# OPTIMASI JUMLAH PRODUKSI METAL ROOF MENGGUNAKAN ALGORITME GENETIKA (STUDI KASUS: PT. COMTECH METALINDO TERPADU)

**SKRIPSI**

Untuk memenuhi sebagian persyaratan

Memperoleh gelar Sarjana Komputer

Disusun oleh:
Febri Ramadhani
NIM: 145150201111164

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

# PENGESAHAN

OPTIMASI JUMLAH PRODUKSI METAL ROOF MENGGUNAKAN ALGORITME GENETIKA (STUDI KASUS: PT. COMTECH METALINDO TERPADU)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Febri Ramadhani
NIM: 145150201111164

Skripsi ini telah diuji dan dinyatakan lulus pada
30 Juli 2018
Telah diperiksa dan disetujui oleh:

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| Budi Darma Setiawan, S.Kom, M.Cs | Candra Dewi, S.Kom, M.Sc |
| NIP: 198410152014041002 | NIP: 19771114 200312 2 001 |

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

IDENTITAS TIM PENGUJI

PENGUJI 1

Indriati, S.T, M.Kom
NIP/NIK. 19831013 201504 2 002

PENGUJI 2

Mochammad Ali Fauzi, S.Kom, M.Kom
NIP/NIK. 201502 890101 1 001

# PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 30 Juli 2018


Febri Ramadhani

NIM: 145150201111164

# KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT atas segala Rahmat dan Karunia-Nya, sehingga skripsi peneitian yang berjudul "Optimasi Jumlah Produksi Metal Roof Menggunakan Algoritme Genetika (Studi Kasus: PT. Comtech Metalindi Terpadu)" dapat terselesaikan dengan baik.

Saya sebagai penulis menyampaikan rasa terima kasih kepada berbagai pihak karena dalam penyusunan dan penelitian skripsi ini tidak lepas dari bantuan dan dukungan baik yang mereka berikan, diantaranya :

1. Kedua orang tua saya beserta kakak-kakak saya tercinta yang selalu memberikan dukungan dari segi manapun selama proses belajar di bangku perkuliahan Universitas Brawijaya, serta semua kerabat dan saudara yang telah mendoakan.
2. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Ketua Fakultas Ilmu Komputer Universitas Brawijaya.
3. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak M. Tanzil Furqon, S.Kom, M.CompSc. selaku Sekretaris Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
5. Bapak Agus Wahyu Widodo, S.T., M.Cs selaku Kepala Prodi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
6. Bapak Budi Darma Setiawan, S.Kom, M.Cs selaku Dosen Pembimbing I yang telah meluangkan waktu untuk membimbing dan mengarahkan penulis dalam penulisan skripsi dengan penuh perhatian dan kesabaran.
7. Ibu Candra Dewi, S.Kom, M.Sc selaku Dosen Pembimbing II yang telah meluangkan waktu untuk membimbing dan mengarahkan penulis dalam penulisan skripsi dengan penuh perhatian dan kesabaran.
8. Seluruh Dosen Fakultas Ilmu Komputer, Universitas Brawijaya atas ilmu yang bermanfaat bagi penulis.
9. Keluarga Besar HMR Ficry Agam, Riyan ilmi, Riadhi Wicaksana, Rusdan Abdillah, Valdi Cahya, Vandi Cahya, Kefas Tarigan, dan Aulia Akbar Akmal yang sudah banyak membantu membangunkan saya selama di Malang.
10. Keluarga Habis Makan Minum Cengir, Simul, Apoy, Dedek, Iqbal, Aidel, Iqok, Dino, Dega, Toni, dan Ojan sebagai sahabat-sahabat yang selalu mensupport saya.
11. Kontrakan Second Home Muhammad Nadzir, Doni Putra Purbawa, Risailin Dwi Jaka, dan Winny Ardhian Septiko yang selalu bersedia membantu mengajarkan saya kapanpun ketika saya tidak mengerti dan bersedia menampung saya ketika mengerjakan tugas perkuliahan.
12. Teman-teman CL Hanif, Fitko, Odi, Bli, Dika, Munir, Cakson, Kevin, Faris, Sandi, Rio, Rexa, Ijem, Farhan, Reza, Asroful, Igo, Rizal, Steven, Sam, Kaye, dan Kubam yang telah banyak membantu.
13. Teman-Teman HML Informatika yang banyak membantu saya dalam perkuliahan mulai sejak awal saya di Malang.

14. Teman-Teman Informatika angkatan 2014 yang tidak dapat saya sebutkan satu persatu.

Malang, 30 Juli 2018

Febri Ramadhani

# ABSTRAK

Industri manufaktur di Indonesia terus mengalami peningkatan, terutama dalam industri barang cetakan. PT. Comtech Metalindo Terpadu adalah salah satu perusahaan industri barang cetakan yang berlokasi di Kota Pekanbaru. Perusahaan tersebut merupakan perusahaan industri yang memproduksi metal roof. Metal roof dicetak menggunakan bahan baku *Prepainted Galvalum (PPGL)* atau yang lebih sering disebut dengan coil, bahan baku tersebut diimpor dari luar Indonesia. Pemesanan bahan baku membutuhkan waktu 2 bulan hingga bahan baku tersebut tiba. Terdapat 3 jenis produk metal roof yang dijual yaitu spandek, zigzag dan zigzag pasiran. Ketiga item tersebut memiliki komposisi bahan baku, serta memberikan keuntungan yang berbeda. Mengatur jumlah produksi yang tepat merupakan hal yang harus diperhitungkan pemilik perusahaan agar didapatkan keuntungan yang optimal. Berdasarkan permasalahan tersebut untuk mendapatkan jumlah produksi yang tepat atas penggunaan sisa bahan baku, dibutuhkan pengoptimalan jumlah produksi metal roof berdasarkan permintaan yang ada serta stok bahan baku yang tersisa. Optimasi digunakan untuk mengatur jumlah produksi yang ada, sehingga bahan baku yang tersisa dapat digunakan secara optimal serta memberikan keuntungan yang optimal pula. Algoritme Genetika digunakan untuk melakukan optimasi dari 3 gen yang mewakili setiap produk yang ada. Nilai dari gen tersebut merepresentasikan nilai asli dari permintaan yang ada dengan tipe *integer*. Pada reproduksi metode *crossover* yang digunakan adalah *extended intermediate crossover*, sedangkan mutasi dilakukan dengan membangkitkan ulang nilai gen dari suatu kromosom yang dipilih secara acak. Untuk proses seleksi digunakan *elitism selection* untuk menyaring individu yang terbaik, serta digunakan metode *random injection* untuk mencegah konvergensi dini. Berdasarkan pengujian parameter yang telah dilakukan dengan 5 kali percobaan setiap parameter didapatkan ukuran populasi terbaik sebanyak 90, kombinasi *cr*=0.1 dan *mr*=0.9, serta jumlah generasi terbaik sebesar 225 dengan rata-rata nilai *fitness* 7.12126.

Kata Kunci: Optimasi, Produksi, Algoritme Genetika, Metal Roof

# ABSTRACT

*Manufacturing industry in Indonesia continues to increase, especially in the molding industry. PT. Comtech Metalindo Terpadu is one of molded goods industry company located in Pekanbaru City. The company is an industrial company that produces metal roof. The metal roof is printed using Prepainted Galvalum (PPGL) raw material or more commonly referred to as coil, the raw material is imported from other countries. The ordering of raw materials takes 2 months until the raw material arrives. There are 3 types of metal roof products sold are spandek, zigzag and zigzag charcoal. All three items have the composition of raw materials, as well as providing benefits that are different. Setting the right amount of production is the thing that must be taken into account by the owner of the company in order to obtain optimal benefits. Based on these problems to get the right amount of production on the use of the remaining raw materials, it is necessary to optimize the number of metal roof production based on the existing demand and the remaining stock of raw materials. Optimization is used to regulate the amount of existing production so that the remaining raw materials can be used optimally and provide optimal benefits as well. Genetic Algorithms are used to optimize the 3 genes that represent each product. The value of the gene represents the original value of the existing query with the integer type. In the reproduction, the crossover method that used is the extended intermediate crossover. Whereas the mutation is performed by reviving the gene values of a randomly selected chromosome. For the selection process used elitism selection to screen the best individual and used random injection method to prevent early convergence. Based on testing of parameters that have been done with 5 times each parameter is got the best population size 90, the combination of cr = 0.1 and mr = 0.9, and total of best generation equal to 225 with average fitness value 7.12126.*

*Keywords: Optimazion, Production, Genetic Algorithm, Metal Roof*

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# DAFTAR SOURCE CODE

# DAFTAR LAMPIRAN

# BAB 1 PENDAHULUAN

## 1.1 Latar Belakang

Di Indonesia perkembangan industri manufaktur sangat pesat seiring dengan kemajuan teknologi yang ada pada saat ini, kemajuan teknologi tersebut mempermudah proses produksi suatu barang mentah menjadi barang jadi. Industri manufaktur mulai meningkat, terutama pada industri barang cetakan yang impor bahan bakunya terus mengalami peningkatan (Saputra, 2018). Oleh karena itu, para pemilik industri manufaktur terus ditekan untuk memproduksi barang sesuai dengan permintaan yang ada.

Pada industri manufaktur barang cetakan terdapat banyak jenisnya, salah satunya adalah industri barang metal roof. Metal roof merupakan atap metal yang dipasang pada sebuah bangunan, metal roof terbuat dari bahan baku utama yaitu *Prepainted Galvalum (PPGL)* yang diimpor dari luar Indonesia. Metal roof lebih banyak digunakan karena bebannya yang relatif ringan sehingga rangka atap bawah suatu bangunan bisa lebih sederhana. Oleh karena itu, anggaran biaya dalam pendirian sebuah bangunan dapat diminimalkan namun tetap memberikan kualitas yang maksimal.

Berdasarkan hasil wawancara dengan pemilik PT. Comtech Metalindo Terpadu, perusahaan memiliki 3 jenis produk metal roof yang dijual yaitu spandek, zigzag dan zigzag pasiran. Ketiga item tersebut memiliki komposisi bahan baku, serta memberikan keuntungan yang berbeda pula. Dengan perbedaan komposisi bahan baku dan keuntungan suatu item, perusahaan tentunya harus mengatur jumlah produksi agar bahan baku tidak tersisa dan memberikan keuntungan yang baik. Apabila jumlah produksi per item hanya sekedar perkiraan secara subjektif tentunya tidak akan memberikan hasil yang optimal. Oleh karena itu, optimasi dibutuhkan untuk mendapatkan jumlah produksi yang optimal menghabiskan bahan baku yang ada. Sehingga nantinya optimasi memberikan keuntungan yang optimal pula.

Adapun penelitian sejenis yang terkait optimasi peningkatan keuntungan produksi yang dilakukan (Fitrianur, et al., 2017). Dalam penelitian itu melakukan optimasi peningkatan laba produksi abon menggunakan Algoritme Genetika pada UKM Poklahsar Berkah Lumintu-Tulungagung. Setelah penelitian tersebut dilakukan didapatkan hasil *fitness* terbaik adalah 50 dengan nilai ukuran populasi = 100, ukuran generasi = 90, *cr* = 0.8, dan *mr* = 0.2. Penelitian yang juga berkaitan dengan penilitian ini adalah optimasi penjadwalan mata kuliah yang dilakukan oleh Entot Suhartono dalam (Suhartono, 2015), penelitian tersebut menggunakan Algoritme Genetika untuk mendapatkan optimasi dari penjadwalan mata kuliah.

Berdasarkan permasalahan diatas maka penelitian ini mengimplementasikan metode Algoritme Genetika dengan pada optimasi jumlah produksi metal roof. Namun metode Algoritme Genetika terkadang terjadi konvergensi dini, pada kondisi ini hampir semua individu sudah bernilai sama namun solusi yang mendekati optimum belum terpenuhi. Oleh karena itu, dibutuhkan penanganan

konvergensi dini menggunakan *random injection* agar bisa didapatkan hasil yang paling mendekati optimal. Pada penelitian ini berfokus pada penggunaan bahan baku dari produksi metal roof agar dapat mengetahui jumlah produksi sehingga memberikan keuntungan yang mendekati optimal. Diharapkan dari penelitian ini dapat membantu perusahaan untuk mendapatkan gambaran jumlah produksi yang mendekati optimal dalam proses produksi metal roof.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang masalah diatas, maka dapat dibuat rumusan masalah sebagai berikut :

1. Berapa nilai parameter yang mendekati optimal dari implementasi metode Algoritme Genetika pada permasalahan optimasi jumlah produksi metal roof?
2. Bagaimana pengaruh *random injection* untuk menangani konvergensi dini pada permasalahan metal roof menggunakan Algoritme Genetika?

## 1.3 Tujuan

Adapun tujuan penelitian ini adalah sebagai berikut :

1. Mengetahui nilai parameter yang mendekati optimal dari implementasi metode Algoritme Genetika pada optimasi jumlah produksi metal roof.
2. Mengetahui pengaruh *random injection* untuk menangani konvergensi dini pada permasalahan metal roof menggunakan Algoritme Genetika.

## 1.4 Manfaat

Adapun maanfat penelitian ini adalah sebagai berikut :

1. Membantu pihak PT. Comtech Metalindo Terpadu untuk mengoptimalkan penggunaan sisa bahan baku yang ada.
2. Membantu pihak PT. Comtech Metalindo Terpadu untuk memberikan gambaran jumlah produksi yang mendekati optimal agar mendapatkan keuntungan yang optimal.

## 1.5 Batasan Masalah

Batasan masalah pada penelitian ini adalah :

1. Optimasi produksi metal roof menggunakan sisa bahan baku dari bulan sebelumnya untuk mendapatkan jumlah produksi yang optimal pada bulan kedepannya.
2. Optimasi produksi metal roof menggunakan permintaan terendah dan permintaan tertinggi dari permintaan produksi selama sebulan pada bulan februari 2017.
3. Data yang digunanakan merupakan data dari PT. Comtech Metalindo Terpadu.

## 1.6 Sistematika Pembahasan

**BAB I – PENDAHULUAN**

Dalam pendahuluan memuat latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika pembahasan dari penelitian.

**BAB II – LANDASAN KEPUSTAKAAN**

Memuat kajian pustaka dan dasar teori yang berkaitan serta menunjang proses penelitian.

**BAB III – METODOLOGI DAN PERANCANGAN ALGORITME**

Memuat alur kerja yang dilakukan oleh peneliti dalam menyelesaikan permasalahan penelitian.

**BAB IV – IMPLEMENTASI**

Membahas seputar perancangan proses untuk menyelesaikan permasalahan , dan implementasi dari perancangan proses yang telah dibuat.

**BAB V – PENGUJIAN DAN ANALISIS**

Memuat hasil pengujian ukuran populasi, pengujian kombinasi *cr* dan *mr*, pengujian jumlah generasi, dan pengujian *random injection* disertai dengan analisis setiap pengujian.

**BAB VI – PENUTUP**

Memuat kesimpulan dari penelitian yang telah dilakukan dan saran untuk pengembangan penelitian selanjutnya.

# BAB 2 LANDASAN KEPUSTAKAAN

## 2.1 Kajian Pustaka

Terdapat beberapa penelitian sebelumnya yang memiliki keterkaitan dengan permasalahan penelitian ini. Pada kajian pustaka ini akan membahas perbedaan dan relevansi dari penelitian-penelitan tersebut dengan penelitian ini. Penulis mengambil 3 hasil penelitian yang dijadikan kajian pustaka.

Penelitian pertama oleh Fitrianur, Putri dan Wicaksono (2017) yaitu optimasi peningkatan laba produksi abon. Penelitian tersebut memiliki tujuan untuk dapat menghitung optimasi laba secara cepat dan akurat sehingga laba produksi bisa dinaikkan dengan prinsip meminimalkan modal. Penelitian ini menggunakan Algoritme Genetika dengan menggunakan 4 gen pada setiap kromosom yang mewakili produk abon. Penerapan Algoritme Genetika pada dalam penelitian peningkatan laba produksi abon menunjukkan hasil *fitness* terbaik adalah 50 dengan nilai parameter terbaik yakni ukuran populasi = 100, ukuran generasi = 90, *cr* = 0.8, dan *mr* = 0.2.

Penelitian kedua oleh Entot Suhartono (2015) yaitu optimasi penjadwalan mata kuliah. Pada penelitian ini menggunakan Algoritme Genetika. Permasalahan yang ditangani pada penelitian ini adalah pengelolaan komponen-konponen jadwal, seperti: data dosen, mahasiswa, ruangan, mata kuliah, dan waktu perkuliahan dan pembuatan jadwal penggunaan ruang laboratorium dan mata kuliah praktikum komputer. Hasil penelitian ini menggunakan Algoritme Genetika akan selalu menunjukkan kenaikan fitness atau dengan kata lain generasi selanjutnya lebih baik atau minimal sama dengan generasi sebelumnya.

Penelitian ketiga oleh Ilmi, Mahmudy dan Ratnawati (2015) yaitu Optimasi penjadwalan perawat. Pada penelitian ini digunakan representasi permutasi bilangan integer dengan panjang kromosom 360 yang setiap angka pada gennya merepresentasikan nomor id perawat. Metode *crossover* yang digunakan yaitu *one cut-point crossover*, metode mutasi *reciprocal exchange mutation* dan diseleksi dengan elitism *selection*. Penerapan Algoritme Genetika pada dalam penelitian penjadwalan perawat menunjukkan hasil ukuran populasi = 200, ukuran generasi = 150, *cr* = 0.5, dan *mr* = 0.5.

## 2.2 Dasar Teori

### 2.2.1 Produksi PT. Comtech Metalindo Terpadu

Produksi adalah suatu kegiatan yang menghasilkan atau menambah nilai guna suatu barang maupun jasa, kegiatan produksi biasanya dilakukan oleh seorang produsen. Produksi dalam arti lain juga dapat didefinisikan sebagai suatu kegiatan yang dilakukan untuk memenuhi kebutuhan manusia dengan menghasilkan barang maupun jasa. Di dalam ekonomi, produksi adalah suatu kegiatan untuk menciptakan dan penambahan kegunaan suatu barang ataupun jasa (Situmorang, 2008).

PT. Comtech Metalindo Terpadu adalah sebuah perusahaan yang bergerak pada bidang industri manufaktur barang cetakan. Perusaahan ini berlokasi di kota Pekanbaru, Riau. Perusahaan ini memproduksi genteng yang terbuat dari metal atau yang sering disebut dengan metal roof. Metal roof yang diproduksi terdapat 3 jenis produk, yaitu Spandek, Zigzag, dan Zigzag Pasiran. Ketiga produk tersebut diproduksi dari satu bahan baku utama yang diimpor dari luar indonesia, bahan baku tersebut adalah *Prepainted Galvalum (PPGL)* atau yang sering disebut dengan *coil*. Untuk mendapatkan bahan baku utama tersebut perusahaan harus menunggu kurang lebih dalam waktu 2 bulan hingga barang tersebut tiba dan dapat diolah untuk proses produksi. Sehingga, untuk memproduksi produksi harus membutuhkan waktu. Proses produksi metal roof disini menggunakan mesin pencetak untuk mengolah bahan baku *coil* yang ada. Produksi pada PT. Comtech Metalindo Terpadu tidak sepenuhnya menggunakan mesin, masih terdapat tenaga kerja manusia sebagai pembantu proses produksi suatu produk.

### 2.2.2 Optimasi

Optimasi adalah suatu proses yang memiliki tujuan untuk menemukan solusi terbaik dari beberapa solusi yang ada terdapat pada suatu permasalahan bidang ilmu komputer dan matematika Dalam bahasa yang lebih formal, optimasi merupakan suatu proses menemukan solusi terbaik pada fungsi objektif yang memiliki nilai minimal dan nilai maksimal yang berada dalam daerah solusi (Black, 2001).

### 2.2.3 Algoritme Genetika

Algoritme genetika adalah suatu teknik penerapan yang mengacu pada genetika alami yang merupakan salah satu cabang Algoritme Evolusi. Untuk menghasilkan suatu solusi yang optimal, proses pencarian pada Algoritme Genetika dilakukan berdasarkan fungsi probabilistik yang berada di antara sejumlah alternatif titik optimal. Algoritme Genetika diperkenalkan oleh John Holland sebagai metode untuk menyelesaikan permasalahan optimasi. Metode pencarian ini meniru proses evolusi biologis untuk menentukan individu berkualitas dalam suatu kawasan yang dinamakan sebagai populasi. Pemilihan individu dari suatu populasi dievaluasi berdasarkan fungsi *fitness.* Optimasi penjadwalan produksi dalam bidang industri manufaktur menggunakan algoritme genetika merupakan salah satu contoh penerapan metode ini (Mahmudy, et al., 2014).

Dalam Algoritme Genetika individu dalam populasi akan saling bersaing untuk sumber daya dengan melakukan reproduksi, proses ini dilakukan untuk memperoleh individu baru yang mempunyai gen lebih baik dari orang tuanya. Kemudian, seluruh individu akan melalui proses genetik yaitu seleksi alam. Pada proses ini individu yang lebih kuat akan bertahan untuk mendapatkan penerus yang lebih baik (Sutojo, et al., 2011). Siklus dari algoritme genetika secara umum adalah :

1. Representasi kromosom.

2. Inisialisasi populasi awal.
3. Menghitung nilai *fitness*.
4. Proses reproduksi dengan *crossover* dan mutasi.
5. Evaluasi.
6. Seleksi individu untuk generasi selanjutnya.

## 2.2.4 Siklus Algoritme Genetika

Algoritme genetika memiliki 4 siklus utama yaitu inisialisasi, reproduksi, evaluasi serta seleksi. Terdapat tambahan yaitu untuk memenuhi kondisi berhenti Algoritme genetika, dan juga karena permasalahan ini terdapat Algoritme Genetika yang menggunakan *real code* maka akan dijelaskan bagaimana untuk mencegah konvergensi dini Algoritme Genetika.

### 2.2.4.1 Inisialisasi

Inisialisasi merupakan proses pembangkitan himpunan solusi baru secara acak yang terdiri atas sejumlah *string* kromosom, pembangkitan ini ditempatkan pada penampungan yang disebut populasi. Dalam tahap ini harus ditentukan ukuran populasi (*popSize*), nilai ini menyatakan jumlah individu/kromosom yang ditampung dalam suatu populasi. Panjang setiap *string* kromosom (*stringLen*) dihitung berdasarkan presisi variabel solusi yang dicari (Mahmudy, 2013).

### 2.2.4.2 Reproduksi

Reproduksi merupakan proses menghasilkan keturunan dari individu-individu yang ada di populasi. Himpunan keturunan ini ditempatkan dalam penampungan yang dinamakan sebagai *offspring*. Terdapat dua operator genetika yang digunakan dalam proses ini, yaitu tukar silang (*crossover*) dan mutasi (*mutation*). Terdapat banyak metode *crossover* dan *mutation* yang telah dikembangkan oleh para peneliti, namun metode tersebut biasanya bersifat spesifik terhadap masalah dan representasi kromosom yang digunakan (Mahmudy, 2013).

1. *Crossover*

   *Crossover* merupakan operator algoritme genetika yang membutuhkan parameter dua buah kromosom induk untuk menghasilkan keturunan nya. Operator ini akan menghasilkan dua buah kromosom baru, yang disebut dengan *child* atau kromosom anak. Metode *extended intermediate crossover* (Muhlenbein & Schlierkamp-Voosen, 1993) akan menghasilkan keturunan pada *offspring* dari kombinasi nilai dua induk. Misal P1 dan P2 adalah dua induk untuk *crossover*, serta C1 dan C2 merupakan anak yang dihasilkan dari *crossover* ini. Maka, proses reproduksi menggunakan metode *extended intermediate crossover* dapat dilakukan dengan menggunakan Persamaan 2.1 dan Persamaan 2.2.

$$C1 = P1 + \alpha\,(P2 - P1) \quad (2.1)$$

$$C2 = P2 + \alpha\,(P1 - P2) \quad (2.2)$$

   Pada Persamaan 2.1 dan Persamaan 2.2 diatas nilai α dipilih secara acak pada interval [-0,25, 1,25].

2. Mutasi

Mutasi merupakan proses mengubah nilai dari satu atau beberapa gen dalam suatu kromosom dan setelah nilai tersebut diubah, kromosom tersebut akan menjadi suatu kromosom baru yang ditempatkan pada *offspring*. Mutasi ini berperan untuk menggantikan gen yang hilang dari populasi akibat proses seleksi. Sehingga mutasi memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi (Kusumadewi, 2003). Mutasi yang digunakan pada penelitian ini adalah dengan cara memilih kromosom secara *random* lalu memilih satu gen secara *random*, dan kemudian mengubah nilai pada gen tersebut (Mahmudy, 2013).

### 2.2.4.3 Evaluasi

Evaluasi merupakan penggabungan seluruh individu yang ada dari populasi dan *offspring* hasil reproduksi, untuk menghitung nilai kebugaran (*fitness*) setiap individu. Semakin besar nilai *fitness* suatu individu maka akan semakin baik individu tersebut untuk dijadikan sebagai calon solusi (Mahmudy, 2013). Individu dievaluasi berdasarkan fungsi tertentu sebagai ukuran kinerjanya, individu dengan nilai *fitness* tinggi pada kromosomnya yang akan dipertahankan. Sedangkan, individu yang bernilai *fitness* rendah akan dibuang. Perhitungan fungsi *fitness* dapat dibentuk sesuai dengan permasalahan yang ada, sehingga akan menghasilkan solusi yang sesuai dengan permasalahan tersebut.

### 2.2.4.4 Seleksi

Seleksi dilakukan dengan memilih individu dari himpunan populasi dan *offspring* untuk dipertahankan hidup pada generasinya. Semakin besar nilai *fitness* sebuah individu, maka semakin besar peluang individu tersebut untuk dipertahankan. Hal tersebut dilakukan agar generasi berikutnya lebih baik dari generasi yang sekarang. Metode seleksi yang digunakan pada penelitian ini adalah seleksi *elitism*. Metode ini mengumpulkan semua individu populasi dan *offspring* dalam satu penampungan. Lalu, individu sebanyak ukuran populasi dalam penampungan ini akan lolos untuk masuk dalam generasi selanjutnya sebagai hasil seleksi. Metode seleksi ini menjamin individu yang terbaik akan selalu lolos dikarenkan (Mahmudy, 2013). *Pseudo-code elitism selection* disajikan pada *Source Code* 2.1.

*Pseudo-Code Elitism Selection*

```
1   PROCEDURE ElitismSelection
2   Input:
3       POP:        himpunan individu pada populasi
4       pop_size:   ukuran populasi
5       OS:         himpunan individu anak (offspring) hasil reproduksi
6                   menggunakan crossover and mutasi
7
8   Output:
9       POP:       himpunan individu pada populasi setelah proses seleksi
10                 selesai
11
12  /* gabungkan individu pada POP dan OS ke dalam TEMP  */
13  TEMP  ← Merge (POP,OS)
14  /* urutkan individu berdasarkan fitness secara ascending */
```

```
15  OrderAscending (Temp)
16  /* copy pop_size individu terbaik ke POP */
17  POP ← CopyBest (Temp, pop_size)
18
19  END PROCEDURE
```

***Source Code* 2.1 *Pseudo-Code Elitism Selection***

Berikut adalah langkah-langkah seleksi menggunakan *elitism*:

1. Gabungkan populasi awal dengan *offspring* hasil reproduksi beserta nilai *fitnessnya*.
2. Urutkan individu gabungan berdasarkan nilai *fitness* terbesar ke terkecil (*descending*).
3. Ambil individu teratas sebanyak *PopSize* untuk dilanjutkan pada generasi selanjutnya sebagai populasi baru.

#### 2.2.4.5 Penanganan Konvergensi Dini

Konvergensi dini merupakan suatu kondisi dimana himpunan solusi Algoritme Genetika hampir bernilai sama, sehingga Algoritme Genetika tidak dapat mencari himpunan solusi yang lebih luas. Oleh sebab itu dibutuhkan penanganan konvergensi dini agar didapatkan hasil yang paling optimal. Ada banyak metode untuk mengatasi permasalahan konvergensi dini ini. Salah satu metode sederhana yang bisa diterapkan adalah dengan melakukan *random injection*. Metode ini akan membangkitkan ulang nilai setiap gen pada individu sebanyak *n* individu terakhir, yang berarti proses seleksi nya hanya memilih sebanyak *popSize-n* individu terakhir. Kemudian, barulah nilai gen pada *n* individu terakhir tersebut akan dibangkitkan secara random seperti pada saat inisialisasi (Mahmudy, et al., 2013a; Mahmudy, et al., 2013b). Untuk *n* individu terakhir bisa didapatkan dengan menggunakan Persamaan 2.3.

$$n = 0.1 \times Popsize \tag{2.3}$$

Pada umumnya perubahan sebanyak 10% dari *popSize* ini sudah cukup memadai untuk menangani permasalahan konvergensi dini tersebut, sehingga keragaman individu akan tetap terjaga. *Random Injection* ini juga tidak harus diterapkan pada setiap iterasi, karena akan memakan waktu komputasi yang lebih lama.

Dengan mengacu struktur Algoritme Genetika murni maka teknik penanganan konvergensi dini dengan *random injection* bisa disusun sesuai *Source Code* 2.2.

*Pseudo-Code Random Injection*

```
1   procedure AlgoritmaGenetika
2   begin
3      t = 0
4      inisialisasi P(t)
5      while (bukan kondisi berhenti) do
6         reproduksi C(t) dari P(t)
7         evaluasi P(t) dan C(t)
8         seleksi P(t+1) dari P(t) dan C(t)
9         if (t mod g = 0)
10        replace n individu
```

```
11          end
12          t = t + 1
13      end while
14  end
```

***Source Code* 2.2 *Pseudo-Code Random Injection***

#### 2.2.4.6 Kondisi Berhenti

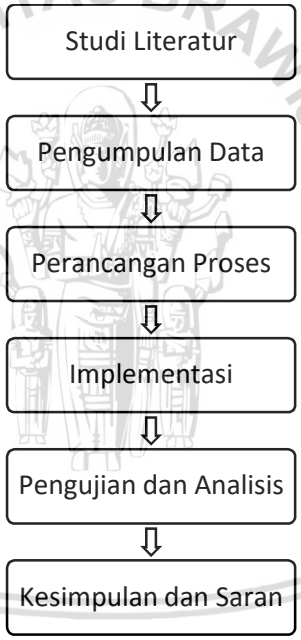
Kondisi berhenti merupakan kondisi akhir dari proses Algoritme Genetika, terdapat banyak pilihan untuk memenuhi kondisi berhenti ini. Salah satu kondisi berhenti adalah iterasi berhenti sampai generasi *n*. Nilai *n* ditentukan sebelumnya berdasarkan beberapa eksperimen pendahuluan. Semakin tinggi kompleksitas suatu permasalahan maka nilai tentunya nilai *n* semakin besar. Oleh karena itu, nilai *n* ditentukan sedemikian rupa sehingga didapatkan konvergensi yang optimal dan tidak terdapat solusi yang lebih baik (Yogeswaran, et al., 2009).

# BAB 3 METODOLOGI DAN PERANCANGAN ALGORITME

Pada bab 3 metodologi dan perancagan algoritme ini menjelaskan bagaimana langkah-langkah atau metodologi yang dipergunakan untuk mencapai tujuan penelitian, serta pada bab ini juga akan menjelaskan tentang persiapan penyelesaian masalah dan perancangan algoritme untuk menyelesaikan permasalahan pada penelitian ini.

## 3.1 Metodologi Penelitian

Metodologi penelitian ini menjelaskan langkah-langkah yang dipergunakan untuk mencapai tujuan penelitian, yaitu studi literatur, pengumpulan data, perancangan proses, implementasi, pengujian dan analisis, lalu terakhir kesimpulan dan saran. Berikut ini merupakan diagram alir dalam penelitian seperti pada Gambar 3.1.




**Gambar 3.1 Diagram Alir Metodologi Penelitian**

### 3.1.1 Studi Literatur

Tahapan pertama dalam penelitian yang dilakukan peneliti adalah studi literature atau studi kepustakaan. Studi Literatur merupakan proses mengumpulkan sumber-sumber yang berhubungan dengan topik penelitian. Dengan adanya studi kepustakaan diharapkan bisa digunakan untuk memperkaya pengetahuan akan teori yang digunakan peneliti dalam penelitian. Studi kepustakaan yang digunakan dalam penelitian ini adalah sebagai berikut:

- Produksi PT. Comtech Metalindo Terpadu
- Optimasi
- Algoritme Genetika

- Siklus Algoritme Genetika

### 3.1.2 Pengumpulan Data

Pada tahapan ini akan menjelaskan bagaimana metode pengumpulan data yang dilakukan berdasarkan permasalahan pada penelitian yang diangkat penulis. Penulis menggunakan data kuantitatif (data yang dinyatakan dalam bentuk angka), data tersebut merupakan data yang diperoleh dari PT. Comtech Metalindo Terpadu yang dimana data tersebut berisi parameter-parameter yang akan digunakan untuk menyelesaikan permasalahan pada penelitian ini. Parameter yang digunakan dari 3 item yang dijual adalah stok bahan baku, laba per produk, komposisi bahan baku untuk memproduksi per produk, serta data permintaan. Setelah semua data terkumpul, penulis nantinya akan membuat tabel data supaya mempermudah dalam memahami datanya.

### 3.1.3 Perancangan Proses

Pada tahapan ini akan dijelaskan perancangan proses untuk menyelesaikan permasalahan pada penelitian yang diangkat penulis. Dalam perancangan proses dilakukan pembuatan arsitektur jumlah produksi metal roof menggunakan algoritme genetika dengan melalui tahapan proses optimasi. Adapun beberapa proses yang dilakukan pada perancangan ini adalah persiapan data, perancangan Algoritme Genetika untuk menyelesaikan permasalahan, perhitungan manual Algoritme Genetika, serta perancangan pengujian yang akan dilakukan nantinya. Perancangan proses ini akan berisi alur penyelesaian masalah optimasi menggunakan Algoritme Genetika dari mulai proses awal hingga proses berhenti untuk mendapatkan hasil solusi yang mendekati optimal. Tahapannya dimulai dengan proses inisialisasi kromosom untuk memulai proses optimasi, dan selanjutnya dilakukan *crossover* dan mutasi untuk reproduksi. Setelah dilakukan reproduksi akan dievaluasi untuk dilakukan seleksi yang terpilih untuk menjadi kromosom pada generasi selanjutnya. Kemudian generasi akan terus dilanjutkan sampai hasil tidak mengalami penurunan lagi atau hasil mendekati hasil optimum.

### 3.1.4 Implementasi

Pada tahapan ini akan dilakukan implementasi sesuai dengan apa yang telah dibuat dalam perancangan proses sebelumnya. Dalam impementasi ini menerapkan metode penelitian yang sudah dipilih dalam menyelesaikan permasalahan yang diangkat. Dalam implementasi ini penulis menggunakan bahasa pemrograman *Java*. Lalu untuk membangun implementasi dari optimasi jumlah produksi metal roof menggunakan Algoritme Genetika sesuai dengan perancangan proses yang telah dibuat. Pada implementasi ini menggunakan Algoritme Genetika dengan pengkodean *real* (RCGA), dimana kromosom terdiri dari 3 gen yang mewakili 3 produk metal roof yang dijual. Nilai pada gen menggunakan representasi *integer* yang mewakili nilai asli dari jumlah yang akan diproduksi nantinya. Hasil dari implementasi nantinya akan mengeluarkan hasil akhir berupa kombinasi jumlah produksi dari 3 item yang ada.

### 3.1.5 Pengujian dan Analisis

Pada tahapan ini adalah pengujian dari implementasi yang telah dilakukan untuk mengetahui manakah solusi yang mendekati optimum. Pengujian merupakan proses pengolahan data primer yang didapatkan dari PT. Comtech Metalindo Terpadu. Pengujian yang akan dilakukan adalah pengujian untuk mendapatkan ukuran populasi, pengujian kombinasi *cr* dan *mr*, pengujian jumlah generasi, dan pengujian *random injection*. Pengujian dimulai dengan pengujian ukuran populasi yang terbaik, setelah didapatkan nilai populasi yang terbaik akan dilanjutkan dengan pengujian kombinasi *cr* dan *mr*. Pada pengujian ini nilai populasi terbaik akan digunakan sebagai parameter untuk mendapatkan kombinasi *cr* dan *mr* terbaik. Pada pengujian generasi, nilai populasi terbaik serta kombinasi *cr* dan *mr* terbaik dijadikan sebagai parameter pengujian. Pada pengujian generasi inilah nantinya akan diambil hasil yang mendekati optimum dari parameter terbaik. Selanjutnya dilakukan pengujian *random injection*, untuk membandingkan hasil optimasi yang menggunakan *random injection* dan yang tidak menggunakan *random injection*. Setelah semua pengujian dilakukan, nantinya akan dilanjutkan dengan proses analisis. Semua hasil pengujian akan dianilisis hasilnya, analisis yang dilakukan akan menjelaskan bagaimana hasil dari setiap pengujian yang telah dilakukan.

### 3.1.6 Kesimpulan dan Saran

Pada tahapan kesimpulan dan saran, setelah mendapatkan hasil dari optimasi yang telah dilakukan tentunya akan menjawab semua rumusan masalah yang ada untuk dirangkum keseluruhannya di dalam kesimpulan. Dan setelah melakukan penelitian ini, peneliti akan memberikan saran yang lebih baik untuk kedepannya.

## 3.2 Perancangan Algoritme

Perancangan algoritme merupakan perancangan dari proses, bagaimana algoritme dapat menyelesaikan permasalahan yang ada. Adapun perancangan algoritme yang dilakukan yaitu, persiapan data, representasi kromosom, perhitungan *fitness*, perancangan penyelesaian masalah, perhitungan manual, perancangan antarmuka, dan perancangan pengujian.

### 3.2.1 Persiapan Data

Permasalahan yang akan diselesaikan pada penelitian ini adalah masalah optimasi jumlah produksi metal roof pada studi kasus PT. Comtech Metalindo Terpadu. Metal roof yang diproduksi terdiri dari 3 jenis produk yaitu spandek, zigzag dan zigzag pasiran. Selama ini dalam proses produksi barang perusahaan hanya menggunakan perkiraan secara subjektif dalam menentukan jumlah produksi tanpa memperhatikan penggunaan bahan baku yang optimal sehingga terkadang keuntungan yang didapatkan tidak bisa maksimal dari penggunaan bahan baku yang ada. Dari permasalahan tersebut, diperlukan sebuah sistem yang mampu menghitung optimasi jumlah produksi barang dengan memperhatikan penggunaan bahan baku yang optimal agar bahan baku yang ada dapat digunakan

# BAB 1 PENDAHULUAN

## 1.1 Latar Belakang

Di Indonesia perkembangan industri manufaktur sangat pesat seiring dengan kemajuan teknologi yang ada pada saat ini, kemajuan teknologi tersebut mempermudah proses produksi suatu barang mentah menjadi barang jadi. Industri manufaktur mulai meningkat, terutama pada industri barang cetakan yang impor bahan bakunya terus mengalami peningkatan (Saputra, 2018). Oleh karena itu, para pemilik industri manufaktur terus ditekan untuk memproduksi barang sesuai dengan permintaan yang ada.

Pada industri manufaktur barang cetakan terdapat banyak jenisnya, salah satunya adalah industri barang metal roof. Metal roof merupakan atap metal yang dipasang pada sebuah bangunan, metal roof terbuat dari bahan baku utama yaitu *Prepainted Galvalum (PPGL)* yang diimpor dari luar Indonesia. Metal roof lebih banyak digunakan karena bebannya yang relatif ringan sehingga rangka atap bawah suatu bangunan bisa lebih sederhana. Oleh karena itu, anggaran biaya dalam pendirian sebuah bangunan dapat diminimalkan namun tetap memberikan kualitas yang maksimal.

Berdasarkan hasil wawancara dengan pemilik PT. Comtech Metalindo Terpadu, perusahaan memiliki 3 jenis produk metal roof yang dijual yaitu spandek, zigzag dan zigzag pasiran. Ketiga item tersebut memiliki komposisi bahan baku, serta memberikan keuntungan yang berbeda pula. Dengan perbedaan komposisi bahan baku dan keuntungan suatu item, perusahaan tentunya harus mengatur jumlah produksi agar bahan baku tidak tersisa dan memberikan keuntungan yang baik. Apabila jumlah produksi per item hanya sekedar perkiraan secara subjektif tentunya tidak akan memberikan hasil yang optimal. Oleh karena itu, optimasi dibutuhkan untuk mendapatkan jumlah produksi yang optimal menghabiskan bahan baku yang ada. Sehingga nantinya optimasi memberikan keuntungan yang optimal pula.

Adapun penelitian sejenis yang terkait optimasi peningkatan keuntungan produksi yang dilakukan (Fitrianur, et al., 2017). Dalam penelitian itu melakukan optimasi peningkatan laba produksi abon menggunakan Algoritme Genetika pada UKM Poklahsar Berkah Lumintu-Tulungagung. Setelah penelitian tersebut dilakukan didapatkan hasil *fitness* terbaik adalah 50 dengan nilai ukuran populasi = 100, ukuran generasi = 90, *cr* = 0.8, dan *mr* = 0.2. Penelitian yang juga berkaitan dengan penilitian ini adalah optimasi penjadwalan mata kuliah yang dilakukan oleh Entot Suhartono dalam (Suhartono, 2015), penelitian tersebut menggunakan Algoritme Genetika untuk mendapatkan optimasi dari penjadwalan mata kuliah.

Berdasarkan permasalahan diatas maka penelitian ini mengimplementasikan metode Algoritme Genetika dengan pada optimasi jumlah produksi metal roof. Namun metode Algoritme Genetika terkadang terjadi konvergensi dini, pada kondisi ini hampir semua individu sudah bernilai sama namun solusi yang mendekati optimum belum terpenuhi. Oleh karena itu, dibutuhkan penanganan

konvergensi dini menggunakan *random injection* agar bisa didapatkan hasil yang paling mendekati optimal. Pada penelitian ini berfokus pada penggunaan bahan baku dari produksi metal roof agar dapat mengetahui jumlah produksi sehingga memberikan keuntungan yang mendekati optimal. Diharapkan dari penelitian ini dapat membantu perusahaan untuk mendapatkan gambaran jumlah produksi yang mendekati optimal dalam proses produksi metal roof.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang masalah diatas, maka dapat dibuat rumusan masalah sebagai berikut :

1. Berapa nilai parameter yang mendekati optimal dari implementasi metode Algoritme Genetika pada permasalahan optimasi jumlah produksi metal roof?
2. Bagaimana pengaruh *random injection* untuk menangani konvergensi dini pada permasalahan metal roof menggunakan Algoritme Genetika?

## 1.3 Tujuan

Adapun tujuan penelitian ini adalah sebagai berikut :

1. Mengetahui nilai parameter yang mendekati optimal dari implementasi metode Algoritme Genetika pada optimasi jumlah produksi metal roof.
2. Mengetahui pengaruh *random injection* untuk menangani konvergensi dini pada permasalahan metal roof menggunakan Algoritme Genetika.

## 1.4 Manfaat

Adapun maanfat penelitian ini adalah sebagai berikut :

1. Membantu pihak PT. Comtech Metalindo Terpadu untuk mengoptimalkan penggunaan sisa bahan baku yang ada.
2. Membantu pihak PT. Comtech Metalindo Terpadu untuk memberikan gambaran jumlah produksi yang mendekati optimal agar mendapatkan keuntungan yang optimal.

## 1.5 Batasan Masalah

Batasan masalah pada penelitian ini adalah :

1. Optimasi produksi metal roof menggunakan sisa bahan baku dari bulan sebelumnya untuk mendapatkan jumlah produksi yang optimal pada bulan kedepannya.
2. Optimasi produksi metal roof menggunakan permintaan terendah dan permintaan tertinggi dari permintaan produksi selama sebulan pada bulan februari 2017.
3. Data yang digunanakan merupakan data dari PT. Comtech Metalindo Terpadu.

## 1.6 Sistematika Pembahasan

**BAB I – PENDAHULUAN**

Dalam pendahuluan memuat latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika pembahasan dari penelitian.

**BAB II – LANDASAN KEPUSTAKAAN**

Memuat kajian pustaka dan dasar teori yang berkaitan serta menunjang proses penelitian.

**BAB III – METODOLOGI DAN PERANCANGAN ALGORITME**

Memuat alur kerja yang dilakukan oleh peneliti dalam menyelesaikan permasalahan penelitian.

**BAB IV – IMPLEMENTASI**

Membahas seputar perancangan proses untuk menyelesaikan permasalahan , dan implementasi dari perancangan proses yang telah dibuat.

**BAB V – PENGUJIAN DAN ANALISIS**

Memuat hasil pengujian ukuran populasi, pengujian kombinasi *cr* dan *mr*, pengujian jumlah generasi, dan pengujian *random injection* disertai dengan analisis setiap pengujian.

**BAB VI – PENUTUP**

Memuat kesimpulan dari penelitian yang telah dilakukan dan saran untuk pengembangan penelitian selanjutnya.

# BAB 2 LANDASAN KEPUSTAKAAN

## 2.1 Kajian Pustaka

Terdapat beberapa penelitian sebelumnya yang memiliki keterkaitan dengan permasalahan penelitian ini. Pada kajian pustaka ini akan membahas perbedaan dan relevansi dari penelitian-penelitan tersebut dengan penelitian ini. Penulis mengambil 3 hasil penelitian yang dijadikan kajian pustaka.

Penelitian pertama oleh Fitrianur, Putri dan Wicaksono (2017) yaitu optimasi peningkatan laba produksi abon. Penelitian tersebut memiliki tujuan untuk dapat menghitung optimasi laba secara cepat dan akurat sehingga laba produksi bisa dinaikkan dengan prinsip meminimalkan modal. Penelitian ini menggunakan Algoritme Genetika dengan menggunakan 4 gen pada setiap kromosom yang mewakili produk abon. Penerapan Algoritme Genetika pada dalam penelitian peningkatan laba produksi abon menunjukkan hasil *fitness* terbaik adalah 50 dengan nilai parameter terbaik yakni ukuran populasi = 100, ukuran generasi = 90, *cr* = 0.8, dan *mr* = 0.2.

Penelitian kedua oleh Entot Suhartono (2015) yaitu optimasi penjadwalan mata kuliah. Pada penelitian ini menggunakan Algoritme Genetika. Permasalahan yang ditangani pada penelitian ini adalah pengelolaan komponen-konponen jadwal, seperti: data dosen, mahasiswa, ruangan, mata kuliah, dan waktu perkuliahan dan pembuatan jadwal penggunaan ruang laboratorium dan mata kuliah praktikum komputer. Hasil penelitian ini menggunakan Algoritme Genetika akan selalu menunjukkan kenaikan fitness atau dengan kata lain generasi selanjutnya lebih baik atau minimal sama dengan generasi sebelumnya.

Penelitian ketiga oleh Ilmi, Mahmudy dan Ratnawati (2015) yaitu Optimasi penjadwalan perawat. Pada penelitian ini digunakan representasi permutasi bilangan integer dengan panjang kromosom 360 yang setiap angka pada gennya merepresentasikan nomor id perawat. Metode *crossover* yang digunakan yaitu *one cut-point crossover*, metode mutasi *reciprocal exchange mutation* dan diseleksi dengan elitism *selection*. Penerapan Algoritme Genetika pada dalam penelitian penjadwalan perawat menunjukkan hasil ukuran populasi = 200, ukuran generasi = 150, *cr* = 0.5, dan *mr* = 0.5.

## 2.2 Dasar Teori

### 2.2.1 Produksi PT. Comtech Metalindo Terpadu

Produksi adalah suatu kegiatan yang menghasilkan atau menambah nilai guna suatu barang maupun jasa, kegiatan produksi biasanya dilakukan oleh seorang produsen. Produksi dalam arti lain juga dapat didefinisikan sebagai suatu kegiatan yang dilakukan untuk memenuhi kebutuhan manusia dengan menghasilkan barang maupun jasa. Di dalam ekonomi, produksi adalah suatu kegiatan untuk menciptakan dan penambahan kegunaan suatu barang ataupun jasa (Situmorang, 2008).

PT. Comtech Metalindo Terpadu adalah sebuah perusahaan yang bergerak pada bidang industri manufaktur barang cetakan. Perusaahan ini berlokasi di kota Pekanbaru, Riau. Perusahaan ini memproduksi genteng yang terbuat dari metal atau yang sering disebut dengan metal roof. Metal roof yang diproduksi terdapat 3 jenis produk, yaitu Spandek, Zigzag, dan Zigzag Pasiran. Ketiga produk tersebut diproduksi dari satu bahan baku utama yang diimpor dari luar indonesia, bahan baku tersebut adalah *Prepainted Galvalum (PPGL)* atau yang sering disebut dengan *coil*. Untuk mendapatkan bahan baku utama tersebut perusahaan harus menunggu kurang lebih dalam waktu 2 bulan hingga barang tersebut tiba dan dapat diolah untuk proses produksi. Sehingga, untuk memproduksi produksi harus membutuhkan waktu. Proses produksi metal roof disini menggunakan mesin pencetak untuk mengolah bahan baku *coil* yang ada. Produksi pada PT. Comtech Metalindo Terpadu tidak sepenuhnya menggunakan mesin, masih terdapat tenaga kerja manusia sebagai pembantu proses produksi suatu produk.

### 2.2.2 Optimasi

Optimasi adalah suatu proses yang memiliki tujuan untuk menemukan solusi terbaik dari beberapa solusi yang ada terdapat pada suatu permasalahan bidang ilmu komputer dan matematika Dalam bahasa yang lebih formal, optimasi merupakan suatu proses menemukan solusi terbaik pada fungsi objektif yang memiliki nilai minimal dan nilai maksimal yang berada dalam daerah solusi (Black, 2001).

### 2.2.3 Algoritme Genetika

Algoritme genetika adalah suatu teknik penerapan yang mengacu pada genetika alami yang merupakan salah satu cabang Algoritme Evolusi. Untuk menghasilkan suatu solusi yang optimal, proses pencarian pada Algoritme Genetika dilakukan berdasarkan fungsi probabilistik yang berada di antara sejumlah alternatif titik optimal. Algoritme Genetika diperkenalkan oleh John Holland sebagai metode untuk menyelesaikan permasalahan optimasi. Metode pencarian ini meniru proses evolusi biologis untuk menentukan individu berkualitas dalam suatu kawasan yang dinamakan sebagai populasi. Pemilihan individu dari suatu populasi dievaluasi berdasarkan fungsi *fitness.* Optimasi penjadwalan produksi dalam bidang industri manufaktur menggunakan algoritme genetika merupakan salah satu contoh penerapan metode ini (Mahmudy, et al., 2014).

Dalam Algoritme Genetika individu dalam populasi akan saling bersaing untuk sumber daya dengan melakukan reproduksi, proses ini dilakukan untuk memperoleh individu baru yang mempunyai gen lebih baik dari orang tuanya. Kemudian, seluruh individu akan melalui proses genetik yaitu seleksi alam. Pada proses ini individu yang lebih kuat akan bertahan untuk mendapatkan penerus yang lebih baik (Sutojo, et al., 2011). Siklus dari algoritme genetika secara umum adalah :

1. Representasi kromosom.

2. Inisialisasi populasi awal.
3. Menghitung nilai *fitness*.
4. Proses reproduksi dengan *crossover* dan mutasi.
5. Evaluasi.
6. Seleksi individu untuk generasi selanjutnya.

## 2.2.4 Siklus Algoritme Genetika

Algoritme genetika memiliki 4 siklus utama yaitu inisialisasi, reproduksi, evaluasi serta seleksi. Terdapat tambahan yaitu untuk memenuhi kondisi berhenti Algoritme genetika, dan juga karena permasalahan ini terdapat Algoritme Genetika yang menggunakan *real code* maka akan dijelaskan bagaimana untuk mencegah konvergensi dini Algoritme Genetika.

### 2.2.4.1 Inisialisasi

Inisialisasi merupakan proses pembangkitan himpunan solusi baru secara acak yang terdiri atas sejumlah *string* kromosom, pembangkitan ini ditempatkan pada penampungan yang disebut populasi. Dalam tahap ini harus ditentukan ukuran populasi (*popSize*), nilai ini menyatakan jumlah individu/kromosom yang ditampung dalam suatu populasi. Panjang setiap *string* kromosom (*stringLen*) dihitung berdasarkan presisi variabel solusi yang dicari (Mahmudy, 2013).

### 2.2.4.2 Reproduksi

Reproduksi merupakan proses menghasilkan keturunan dari individu-individu yang ada di populasi. Himpunan keturunan ini ditempatkan dalam penampungan yang dinamakan sebagai *offspring*. Terdapat dua operator genetika yang digunakan dalam proses ini, yaitu tukar silang (*crossover*) dan mutasi (*mutation*). Terdapat banyak metode *crossover* dan *mutation* yang telah dikembangkan oleh para peneliti, namun metode tersebut biasanya bersifat spesifik terhadap masalah dan representasi kromosom yang digunakan (Mahmudy, 2013).

1. *Crossover*

   *Crossover* merupakan operator algoritme genetika yang membutuhkan parameter dua buah kromosom induk untuk menghasilkan keturunan nya. Operator ini akan menghasilkan dua buah kromosom baru, yang disebut dengan *child* atau kromosom anak. Metode *extended intermediate crossover* (Muhlenbein & Schlierkamp-Voosen, 1993) akan menghasilkan keturunan pada *offspring* dari kombinasi nilai dua induk. Misal P1 dan P2 adalah dua induk untuk *crossover*, serta C1 dan C2 merupakan anak yang dihasilkan dari *crossover* ini. Maka, proses reproduksi menggunakan metode *extended intermediate crossover* dapat dilakukan dengan menggunakan Persamaan 2.1 dan Persamaan 2.2.

$$C1 = P1 + \alpha\,(P2 - P1) \quad (2.1)$$

$$C2 = P2 + \alpha\,(P1 - P2) \quad (2.2)$$

   Pada Persamaan 2.1 dan Persamaan 2.2 diatas nilai α dipilih secara acak pada interval [-0,25, 1,25].

2. Mutasi

Mutasi merupakan proses mengubah nilai dari satu atau beberapa gen dalam suatu kromosom dan setelah nilai tersebut diubah, kromosom tersebut akan menjadi suatu kromosom baru yang ditempatkan pada *offspring*. Mutasi ini berperan untuk menggantikan gen yang hilang dari populasi akibat proses seleksi. Sehingga mutasi memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi (Kusumadewi, 2003). Mutasi yang digunakan pada penelitian ini adalah dengan cara memilih kromosom secara *random* lalu memilih satu gen secara *random*, dan kemudian mengubah nilai pada gen tersebut (Mahmudy, 2013).

### 2.2.4.3 Evaluasi

Evaluasi merupakan penggabungan seluruh individu yang ada dari populasi dan *offspring* hasil reproduksi, untuk menghitung nilai kebugaran (*fitness*) setiap individu. Semakin besar nilai *fitness* suatu individu maka akan semakin baik individu tersebut untuk dijadikan sebagai calon solusi (Mahmudy, 2013). Individu dievaluasi berdasarkan fungsi tertentu sebagai ukuran kinerjanya, individu dengan nilai *fitness* tinggi pada kromosomnya yang akan dipertahankan. Sedangkan, individu yang bernilai *fitness* rendah akan dibuang. Perhitungan fungsi *fitness* dapat dibentuk sesuai dengan permasalahan yang ada, sehingga akan menghasilkan solusi yang sesuai dengan permasalahan tersebut.

### 2.2.4.4 Seleksi

Seleksi dilakukan dengan memilih individu dari himpunan populasi dan *offspring* untuk dipertahankan hidup pada generasinya. Semakin besar nilai *fitness* sebuah individu, maka semakin besar peluang individu tersebut untuk dipertahankan. Hal tersebut dilakukan agar generasi berikutnya lebih baik dari generasi yang sekarang. Metode seleksi yang digunakan pada penelitian ini adalah seleksi *elitism*. Metode ini mengumpulkan semua individu populasi dan *offspring* dalam satu penampungan. Lalu, individu sebanyak ukuran populasi dalam penampungan ini akan lolos untuk masuk dalam generasi selanjutnya sebagai hasil seleksi. Metode seleksi ini menjamin individu yang terbaik akan selalu lolos dikarenkan (Mahmudy, 2013). *Pseudo-code elitism selection* disajikan pada *Source Code* 2.1.

*Pseudo-Code Elitism Selection*

```
1   PROCEDURE ElitismSelection
2   Input:
3       POP:        himpunan individu pada populasi
4       pop_size:   ukuran populasi
5       OS:         himpunan individu anak (offspring) hasil reproduksi
6                   menggunakan crossover and mutasi
7
8   Output:
9       POP:       himpunan individu pada populasi setelah proses seleksi
10                 selesai
11
12  /* gabungkan individu pada POP dan OS ke dalam TEMP  */
13  TEMP  ← Merge (POP,OS)
14  /* urutkan individu berdasarkan fitness secara ascending */
```

```
15  OrderAscending (Temp)
16  /* copy pop_size individu terbaik ke POP */
17  POP ← CopyBest (Temp, pop_size)
18
19  END PROCEDURE
```

***Source Code* 2.1 *Pseudo-Code Elitism Selection***

Berikut adalah langkah-langkah seleksi menggunakan *elitism*:

1. Gabungkan populasi awal dengan *offspring* hasil reproduksi beserta nilai *fitnessnya*.
2. Urutkan individu gabungan berdasarkan nilai *fitness* terbesar ke terkecil (*descending*).
3. Ambil individu teratas sebanyak *PopSize* untuk dilanjutkan pada generasi selanjutnya sebagai populasi baru.

#### 2.2.4.5 Penanganan Konvergensi Dini

Konvergensi dini merupakan suatu kondisi dimana himpunan solusi Algoritme Genetika hampir bernilai sama, sehingga Algoritme Genetika tidak dapat mencari himpunan solusi yang lebih luas. Oleh sebab itu dibutuhkan penanganan konvergensi dini agar didapatkan hasil yang paling optimal. Ada banyak metode untuk mengatasi permasalahan konvergensi dini ini. Salah satu metode sederhana yang bisa diterapkan adalah dengan melakukan *random injection*. Metode ini akan membangkitkan ulang nilai setiap gen pada individu sebanyak *n* individu terakhir, yang berarti proses seleksi nya hanya memilih sebanyak *popSize-n* individu terakhir. Kemudian, barulah nilai gen pada *n* individu terakhir tersebut akan dibangkitkan secara random seperti pada saat inisialisasi (Mahmudy, et al., 2013a; Mahmudy, et al., 2013b). Untuk *n* individu terakhir bisa didapatkan dengan menggunakan Persamaan 2.3.

$$n = 0.1 \times Popsize \tag{2.3}$$

Pada umumnya perubahan sebanyak 10% dari *popSize* ini sudah cukup memadai untuk menangani permasalahan konvergensi dini tersebut, sehingga keragaman individu akan tetap terjaga. *Random Injection* ini juga tidak harus diterapkan pada setiap iterasi, karena akan memakan waktu komputasi yang lebih lama.

Dengan mengacu struktur Algoritme Genetika murni maka teknik penanganan konvergensi dini dengan *random injection* bisa disusun sesuai *Source Code* 2.2.

*Pseudo-Code Random Injection*

```
1   procedure AlgoritmaGenetika
2   begin
3      t = 0
4      inisialisasi P(t)
5      while (bukan kondisi berhenti) do
6         reproduksi C(t) dari P(t)
7         evaluasi P(t) dan C(t)
8         seleksi P(t+1) dari P(t) dan C(t)
9         if (t mod g = 0)
10        replace n individu
```

```
11          end
12          t = t + 1
13      end while
14  end
```

***Source Code* 2.2 *Pseudo-Code Random Injection***

#### 2.2.4.6 Kondisi Berhenti

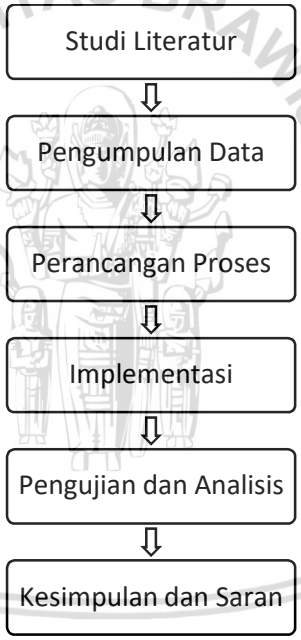
Kondisi berhenti merupakan kondisi akhir dari proses Algoritme Genetika, terdapat banyak pilihan untuk memenuhi kondisi berhenti ini. Salah satu kondisi berhenti adalah iterasi berhenti sampai generasi *n*. Nilai *n* ditentukan sebelumnya berdasarkan beberapa eksperimen pendahuluan. Semakin tinggi kompleksitas suatu permasalahan maka nilai tentunya nilai *n* semakin besar. Oleh karena itu, nilai *n* ditentukan sedemikian rupa sehingga didapatkan konvergensi yang optimal dan tidak terdapat solusi yang lebih baik (Yogeswaran, et al., 2009).

# BAB 3 METODOLOGI DAN PERANCANGAN ALGORITME

Pada bab 3 metodologi dan perancagan algoritme ini menjelaskan bagaimana langkah-langkah atau metodologi yang dipergunakan untuk mencapai tujuan penelitian, serta pada bab ini juga akan menjelaskan tentang persiapan penyelesaian masalah dan perancangan algoritme untuk menyelesaikan permasalahan pada penelitian ini.

## 3.1 Metodologi Penelitian

Metodologi penelitian ini menjelaskan langkah-langkah yang dipergunakan untuk mencapai tujuan penelitian, yaitu studi literatur, pengumpulan data, perancangan proses, implementasi, pengujian dan analisis, lalu terakhir kesimpulan dan saran. Berikut ini merupakan diagram alir dalam penelitian seperti pada Gambar 3.1.




**Gambar 3.1 Diagram Alir Metodologi Penelitian**

### 3.1.1 Studi Literatur

Tahapan pertama dalam penelitian yang dilakukan peneliti adalah studi literature atau studi kepustakaan. Studi Literatur merupakan proses mengumpulkan sumber-sumber yang berhubungan dengan topik penelitian. Dengan adanya studi kepustakaan diharapkan bisa digunakan untuk memperkaya pengetahuan akan teori yang digunakan peneliti dalam penelitian. Studi kepustakaan yang digunakan dalam penelitian ini adalah sebagai berikut:

- Produksi PT. Comtech Metalindo Terpadu
- Optimasi
- Algoritme Genetika

- Siklus Algoritme Genetika

### 3.1.2 Pengumpulan Data

Pada tahapan ini akan menjelaskan bagaimana metode pengumpulan data yang dilakukan berdasarkan permasalahan pada penelitian yang diangkat penulis. Penulis menggunakan data kuantitatif (data yang dinyatakan dalam bentuk angka), data tersebut merupakan data yang diperoleh dari PT. Comtech Metalindo Terpadu yang dimana data tersebut berisi parameter-parameter yang akan digunakan untuk menyelesaikan permasalahan pada penelitian ini. Parameter yang digunakan dari 3 item yang dijual adalah stok bahan baku, laba per produk, komposisi bahan baku untuk memproduksi per produk, serta data permintaan. Setelah semua data terkumpul, penulis nantinya akan membuat tabel data supaya mempermudah dalam memahami datanya.

### 3.1.3 Perancangan Proses

Pada tahapan ini akan dijelaskan perancangan proses untuk menyelesaikan permasalahan pada penelitian yang diangkat penulis. Dalam perancangan proses dilakukan pembuatan arsitektur jumlah produksi metal roof menggunakan algoritme genetika dengan melalui tahapan proses optimasi. Adapun beberapa proses yang dilakukan pada perancangan ini adalah persiapan data, perancangan Algoritme Genetika untuk menyelesaikan permasalahan, perhitungan manual Algoritme Genetika, serta perancangan pengujian yang akan dilakukan nantinya. Perancangan proses ini akan berisi alur penyelesaian masalah optimasi menggunakan Algoritme Genetika dari mulai proses awal hingga proses berhenti untuk mendapatkan hasil solusi yang mendekati optimal. Tahapannya dimulai dengan proses inisialisasi kromosom untuk memulai proses optimasi, dan selanjutnya dilakukan *crossover* dan mutasi untuk reproduksi. Setelah dilakukan reproduksi akan dievaluasi untuk dilakukan seleksi yang terpilih untuk menjadi kromosom pada generasi selanjutnya. Kemudian generasi akan terus dilanjutkan sampai hasil tidak mengalami penurunan lagi atau hasil mendekati hasil optimum.

### 3.1.4 Implementasi

Pada tahapan ini akan dilakukan implementasi sesuai dengan apa yang telah dibuat dalam perancangan proses sebelumnya. Dalam impementasi ini menerapkan metode penelitian yang sudah dipilih dalam menyelesaikan permasalahan yang diangkat. Dalam implementasi ini penulis menggunakan bahasa pemrograman *Java*. Lalu untuk membangun implementasi dari optimasi jumlah produksi metal roof menggunakan Algoritme Genetika sesuai dengan perancangan proses yang telah dibuat. Pada implementasi ini menggunakan Algoritme Genetika dengan pengkodean *real* (RCGA), dimana kromosom terdiri dari 3 gen yang mewakili 3 produk metal roof yang dijual. Nilai pada gen menggunakan representasi *integer* yang mewakili nilai asli dari jumlah yang akan diproduksi nantinya. Hasil dari implementasi nantinya akan mengeluarkan hasil akhir berupa kombinasi jumlah produksi dari 3 item yang ada.

### 3.1.5 Pengujian dan Analisis

Pada tahapan ini adalah pengujian dari implementasi yang telah dilakukan untuk mengetahui manakah solusi yang mendekati optimum. Pengujian merupakan proses pengolahan data primer yang didapatkan dari PT. Comtech Metalindo Terpadu. Pengujian yang akan dilakukan adalah pengujian untuk mendapatkan ukuran populasi, pengujian kombinasi *cr* dan *mr*, pengujian jumlah generasi, dan pengujian *random injection*. Pengujian dimulai dengan pengujian ukuran populasi yang terbaik, setelah didapatkan nilai populasi yang terbaik akan dilanjutkan dengan pengujian kombinasi *cr* dan *mr*. Pada pengujian ini nilai populasi terbaik akan digunakan sebagai parameter untuk mendapatkan kombinasi *cr* dan *mr* terbaik. Pada pengujian generasi, nilai populasi terbaik serta kombinasi *cr* dan *mr* terbaik dijadikan sebagai parameter pengujian. Pada pengujian generasi inilah nantinya akan diambil hasil yang mendekati optimum dari parameter terbaik. Selanjutnya dilakukan pengujian *random injection*, untuk membandingkan hasil optimasi yang menggunakan *random injection* dan yang tidak menggunakan *random injection*. Setelah semua pengujian dilakukan, nantinya akan dilanjutkan dengan proses analisis. Semua hasil pengujian akan dianilisis hasilnya, analisis yang dilakukan akan menjelaskan bagaimana hasil dari setiap pengujian yang telah dilakukan.

### 3.1.6 Kesimpulan dan Saran

Pada tahapan kesimpulan dan saran, setelah mendapatkan hasil dari optimasi yang telah dilakukan tentunya akan menjawab semua rumusan masalah yang ada untuk dirangkum keseluruhannya di dalam kesimpulan. Dan setelah melakukan penelitian ini, peneliti akan memberikan saran yang lebih baik untuk kedepannya.

## 3.2 Perancangan Algoritme

Perancangan algoritme merupakan perancangan dari proses, bagaimana algoritme dapat menyelesaikan permasalahan yang ada. Adapun perancangan algoritme yang dilakukan yaitu, persiapan data, representasi kromosom, perhitungan *fitness*, perancangan penyelesaian masalah, perhitungan manual, perancangan antarmuka, dan perancangan pengujian.

### 3.2.1 Persiapan Data

Permasalahan yang akan diselesaikan pada penelitian ini adalah masalah optimasi jumlah produksi metal roof pada studi kasus PT. Comtech Metalindo Terpadu. Metal roof yang diproduksi terdiri dari 3 jenis produk yaitu spandek, zigzag dan zigzag pasiran. Selama ini dalam proses produksi barang perusahaan hanya menggunakan perkiraan secara subjektif dalam menentukan jumlah produksi tanpa memperhatikan penggunaan bahan baku yang optimal sehingga terkadang keuntungan yang didapatkan tidak bisa maksimal dari penggunaan bahan baku yang ada. Dari permasalahan tersebut, diperlukan sebuah sistem yang mampu menghitung optimasi jumlah produksi barang dengan memperhatikan penggunaan bahan baku yang optimal agar bahan baku yang ada dapat digunakan

dengan sebaiknya dan dapat memberikan keuntungan yang maksimal. Algoritme yang akan digunakan untuk menyelesaikan permasalahan ini adalah algoritme genetika. Data yang digunakan pada penelitian ini adalah data dari PT. Comtech Metalindo Terpadu yang berlokasi di Kota Pekanbaru. Adapun data tersebut adalah jenis produk, laba penjualan per produk, jumlah stok sisa bahan baku, komposisi bahan baku untuk produksi per produk, jumlah permintaan metal roof.

Data komposisi sisa bahan baku per produk untuk produksi dalam satuan kilogram dibutuhkan untuk mengatur penggunaaan sisa bahan baku. Penggunaan sisa bahan baku yang optimal tentunya memerlukan komposisi bahan baku dari setiap produk agar dapat diperhitungkan jumlah produksi yang optimal dari jumlah stok bahan baku yang ada. Data jumlah stok sisa bahan baku merupakan batas penggunaan bahan baku yang harus digunakan untuk optimasi jumlah produksi metal roof. Data laba penjualan per produk dibutuhkan untuk memperhitungkan keuntungan yang didapat dari optimasi jumlah produksi dengan algoritme genetika, sehingga solusi jumlah produksi tetap memberikan keuntungan yang maksimal dari penggunaan sisa bahan baku yang optimal. Sedangkan jumlah permintaan metal roof akan diambil nilai permintaan terendah dan permintaan tertinggi dari produk metal roof yang ada. Jumlah permintaan terendah dan permintaan tertinggi akan digunakan sebagai batasan untuk melakukan *random* nilai gen pada kromosom. Data jumlah permintaan tertinggi dan terendah diperlukan sebagai batasan agar setelah dioptimasi, jumlah produksi metal roof tetap sesuai berdasarkan permintaan pasar. Data yang digunakan pada proses optimasi jumlah produksi metal roof menggunakan Algoritme Genetika ini adalah data produksi pada bulan februari 2017 ditunjukkan pada Tabel 3.1.

**Tabel 3.1 Data Produksi**

| Jenis Produk | Komposisi Bahan Baku(kg) | Laba per Produk | Permintaan terendah | Permintaan tertinggi |
|---|---|---|---|---|
| Spandek | 1,45 | 4500 | 35 | 510 |
| Zigzag | 1,03 | 4750 | 10 | 6000 |
| Zigzag pasiran | 1,03 | 10500 | 1 | 9000 |

Sedangkan untuk stok sisa bahan baku pada bulan februari 2017 adalah 7282 kg.

### 3.2.2 Representasi Kromosom

Pada penelitian ini langkah awal yang dilakukan adalah merepresentasikan kromosom. Representasi kromosom yang digunakan adalah representasi *integer*, representasi *integer* merupakan representasi angka dalam nilai bilangan bulat. Representasi *integer* digunakan karena pada setiap gen dalam kromosom mewakili nilai asli dari jumlah yang akan diproduksi dan juga angka boleh muncul dua kali atau lebih pada gen yang berbeda dalam suatu kromosom. Panjang kromosom yang digunakan adalah 3, karena kromosom mewakili dari 3 jenis

produk yang ada yaitu spandek, zigzag dan zigzag pasiran. Contoh representasi kromosom yang digunakan pada penelitian ini akan dijelaskan pada Tabel 3.2.

**Tabel 3.2 Representasi kromosom**

| **Gen** | x1 | x2 | x3 |
|---|---|---|---|
| **Nilai Gen** | 486 | 4849 | 2324 |

Pada Tabel 3.2 diatas merupakan nilai gen x1, x2, dan x3 yang didapatkan dari nilai *random* jumlah produksi barang spandek, zigzag, dan zigzag pasiran yang bernilai 486, 4849, dan 2324 yang berada dalam rentang permintaan terendah dan permintaan tertinggi yang telah dijelaskan pada Tabel 3.1.

### 3.2.3 *Fitness*

Langkah selanjutnya yang dilakukan setelah adalah desain *fitness* atau perhitungan untuk mendapatkan nilai *fitness*. Nilai *fitness* merupakan faktor penting dalam penentuan solusi pada permasalahan optimasi. Pada penelitian ini nilai fitness dipengaruhi oleh faktor utama yaitu jumlah laba yang akan didapatkan dari jumlah produksi yang ada pada setiap kromosom. Namun, pada permasalahan optimasi ini juga terdapat kendala produksi yang akan dijadikan sebagai penalti. Adapun fungsi penalti tersebut dirumuskan seperti pada Persamaan 3.1.

$$penalti = kendala1 + kendala2 \tag{3.1}$$

Berdasarkan Persamaan 3.1 terdapat 2 kendala yang dijadikan sebagai nilai penalti. Penalti yang tidak boleh dilanggar tersebut adalah jumlah komposisi bahan baku yang digunakan pada solusi jumlah produksi tidak boleh melebihi jumlah stok bahan baku yang ada yang disebut dengan kendala 1. Adapun fungsi kendala 1 bisa diperoleh menggunakan Persamaan 3.2.

$$kendala1 = \begin{cases} 0 & ;1{,}45 * x1 + 1{,}03 * x2 + 1{,}03 * x3 \leq stok\ bahan\ baku \\ 1{,}45 * x1 + 1{,}03 * x2 + 1{,}03 * x3 - stok\ bahan\ baku & ;selainnya \end{cases} \tag{3.2}$$

Persamaan 3.2 adalah nilai kendala 1 yang didapatkan dari komposisi bahan baku tiap produk yang dikalikan dengan nilai pada gen jumlah produksi yang diinisialisaikan sebagai x1, x2 dan x3. Hasil dari komposisi bahan baku yang digunakan pada produksi tersebut tidak boleh melanggar batas stok bahan baku yang ada. Apabila komposisi pada produksi melanggar stok bahan baku yang ada, nilai tersebut akan dimasukkan kedalam nilai penalti.

Sedangkan kendala 2 merupakan nilai gen pada kromosom harus berada pada rentang permintaan terendah dan permintaan tertinggi. Apabila nilai gen berada diluar rentang tersebut, maka nilai tersebut akan dihitung selisihnya dengan batas yang terdekat antara permintaan terendah dan permintaan tertinggi. Lalu, selisih nilai tiap gen akan dijumlahkan dan nilai tersebut akan masuk kedalam nilai kendala 2.

$$selisih1 = \begin{cases} permintaan\ terendah - x1 & ;x1 < permintaan\ terendah \\ x1 - permintaan\ tertinggi & ;x1 > permintaan\ tertinggi \\ 0 & ;permintaan\ terendah \leq x1 \leq permintaan\ tertinggi \end{cases} \tag{3.3}$$

$$selisih2 = \begin{cases} permintaan\ terendah - x2\ ;\ x2 < permintaan\ terendah \\ x2 - permintaan\ tertinggi\ ;\ x2 > permintaan\ tertinggi \\ 0\ ;\ permintaan\ terendah \leq x2 \leq permintaan\ tertinggi \end{cases} \quad (3.4)$$

$$selisih3 = \begin{cases} permintaan\ terendah - A3\ ;\ A3 < permintaan\ terendah \\ A3 - permintaan\ tertinggi\ ;\ A3 > permintaan\ tertinggi \\ 0\ ;\ permintaan\ terendah \leq A3 \leq permintaan\ tertinggi \end{cases} \quad (3.5)$$

Berdasarkan Persamaan 3.3 untuk mendapatkan nilai selisih1 dilakukan pengecekan nilai x1 terhadap nilai rentang batas permintaan tertinggi dan terendah. Jika x1 < permintaan terendah, maka nilai selisih1 akan didapatkan dari permintaan terendah – x1. Jika x1 > permintaan tertinggi, maka nilai selisih1 akan didapatkan dari x1 - permintaan tertinggi. Dan jika x1 berada dalam rentang permintaan terendah dan permintaan tertinggi, maka nilai selisih1 nya adalah 0. Begitu juga untuk Persamaan 3.4 dan Persamaan 3.5, nilai selisih2 dan nilai selisih3 nya didapatkan dengan cara yang sama dilakukan pengecekan nilai x2 dan nilai x3 terhadap nilai rentang batas permintaan tertinggi dan permintaan terendah.

Lalu setelah didapatkan nilai selisih tiap gen, maka akan dijumlahkan untuk didapatkan nilai kendala 2. Untuk lebih detail rumus kendala 2 dijelaskan pada Persamaan 3.6.

$$kendala\ 2 = selisih1 + selisih2 + selisih3 \quad (3.6)$$

Berdasarkan kendala yang ada, fungsi *fitness* dapat dicari menggunakan jumlah laba yang didapatkan dibagi dengan total penalti yang dilanggar. Total penalti yang didapatkan dari kendala 1 dan kendala 2 akan mengurangi nilai *fitness* apabilla total penalti yang dilanggar semakin besar, yang berarti semakin besar total penalti akan mengurangi nilai *fitness* yang ada karena solusi jumlah produksi tersebut melebihi stok bahan baku yang ada dan tidak sesuai dengan permintaan pasar yang ada. Berikut adalah fungsi untuk mendapatkan total laba.

$$total\ laba = 4500 * x1 + 4750 * x2 + 10500 * x3 \quad (3.7)$$

Persamaan 3.7 adalah nilai total laba yang didapatkan dari laba per produk dikalikan dengan nilai pada gen jumlah produksi per produk yang diinisialisasikan masing-masing sebagai x1, x2 dan x3. Oleh karena itu didapatkan fungsi *fitness* pada Persamaan 3.8.

$$fitness = \frac{total\ laba}{(penalti + 1)} \quad (3.8)$$

Persamaan 3.8 adalah fungsi *fitness* yang didapatkan dari nilai total laba di bagi dengan nilai penalti. Namun, nilai penalti bisa terdapat nilai 0 apabila pada kendala tidak melebihi stok bahan baku yang ada dan nilai gen solusi berada dalam rentang permintaan terendah dan permintaan tertinggi. Oleh karena itu, pada nilai penalti ditambahkan nilai 1 untuk mencegah apabila nilai pembagi pada penalti nya bernilai 0. Namun, dikarenakan nilai laba yang terlalu besar, maka nilai *fitness* akan menjadi terlalu besar apabiia dijadikan sebagai hasil *fitness*. Oleh karena itu peneliti menambahkan nilai konstanta agar nilai *fitness* yang didapatkan tidak dalam rentang yang terlalu besar. Konstanta tersebut dijadikan sebagai nilai pembagi pada pada fungsi *fitness* sebelumnya agar hasil *fitness* tersebut tidak

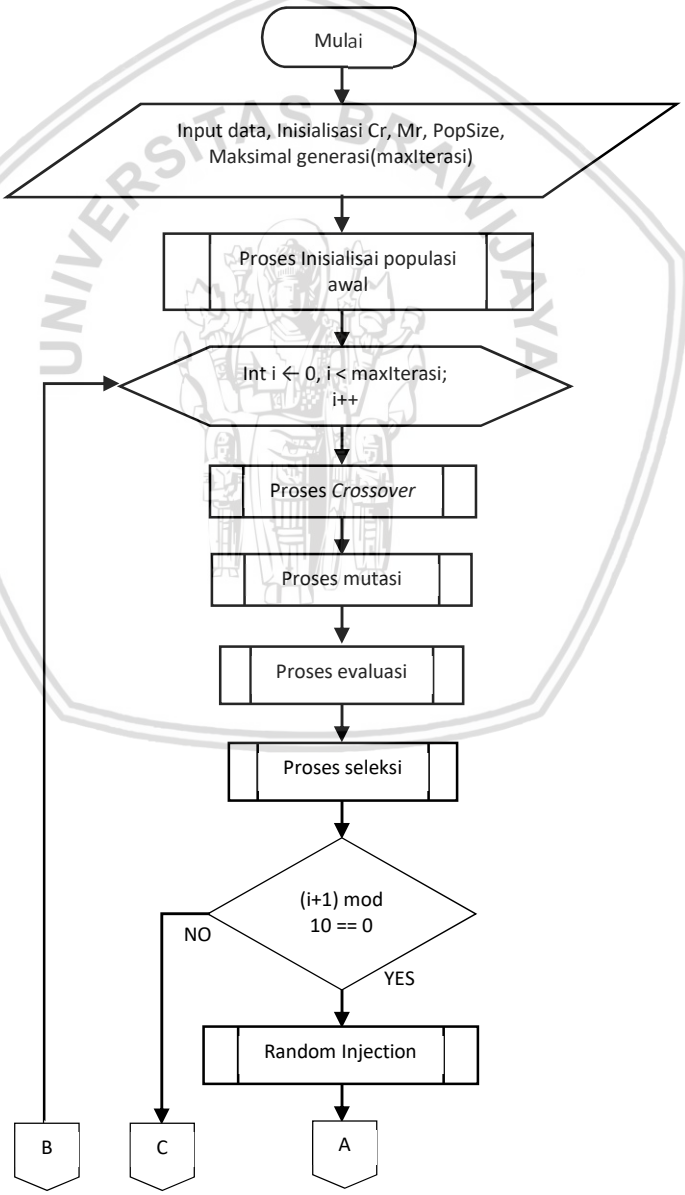
dalam rentang yang terlalu besar. Berikut adalah modifikasi fungsi *fitness* yang akan digunakan pada penelitian ini.

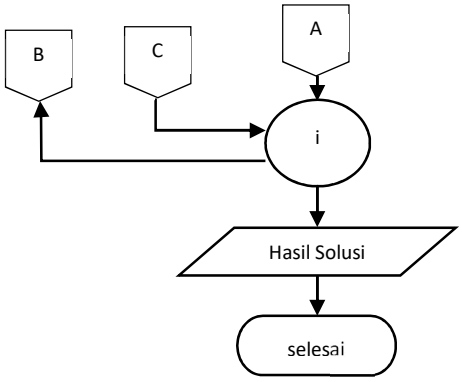
$$fitness = \left(\frac{total\ laba}{(pinalti + 1)}\right) / C \tag{3.9}$$

Persamaan 3.9 diatas adalah fungsi *fitness* yang telah dimodifikasi. Nilai konstanta diinisialisasikan sebagai C, sehingga nilai *fitness* didapatkan dengan nilai total laba dikurangi dengan nilai penalti lalu dibagi dengan nilai konstanta.

### 3.2.4 Perancangan Penyelesaian Masalah

Pada tahap ini peracangan penyelesaian masalah digunakan sebagai dasar untuk membuat implementasi dari optimasi jumlah produksi metal roof menggunakan Algoritme Genetika.



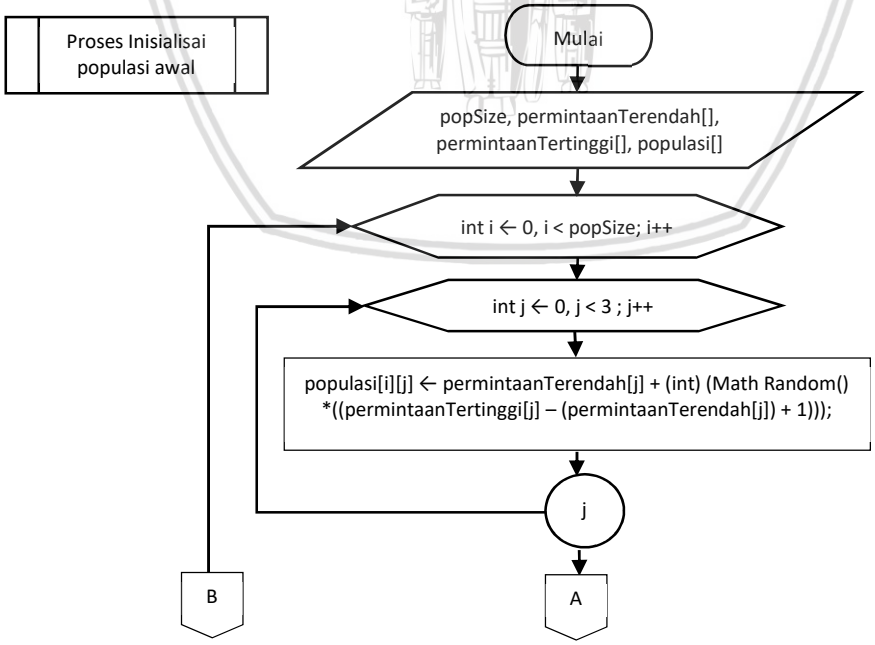


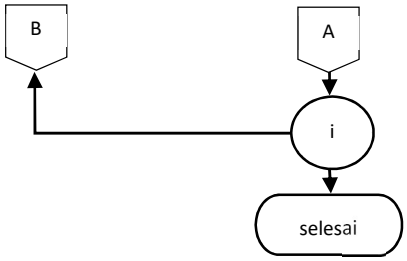

**Gambar 3.2 Diagram Alir Perancangan penyelesaian Masalah**

Berdasarkan Gambar 3.2 Diagram alir penyelesaian masalah terdapat 4 proses utama yang harus dilalui dalam proses algoritme genetika yaitu inisialisasi populasi awal, proses reproduksi yaitu *crossover* dan mutasi, proses perhitungan nilai *fitness* dan evaluasi, proses seleksi, serta proses *random injection*.

### 3.2.4.1 Inisialisasi Populasi Awal

Inisialisasi populasi awal merupakan pembangkitan gen pada individu/kromosom yang diberi nilai *random* sesuai batas permintaan tertinggi dan permintaan terendah, lalu gabungan dari beberapa individu/kromosom inilah yang menjadi populasi awal sesuai dengan ukuran populasi yang akan diinputkan. Masing-masing gen pada setiap individu/kromosom yang ada akan mewakili dari jumlah produksi spandek, zigzag dan zigzag pasiran yang akan menjadi solusi pada optimasi jumlah produksi metal roof. Adapun diagram alir inisialisasi populasi awal dapat dilihat pada Gambar 3.3.






**Gambar 3.3 Diagram Alir Inisialisasi Populasi Awal**

Berdasarkan Gambar 3.3 diagram alir inisialisasi populasi awal terdapat beberapa tahapan proses inisialisasi populasi awal menggunakan algoritme genetika. Dimulai dengan menerima inputan jumlah populasi yang akan dibentuk pada proses, lalu setiap gen dibangkitkan dengan nilai acak dalam rentang diantara nilai permintaan terendah dan nilai permintaan tertinggi masing-masing gen. Setelah nilai acak tiap gen dibangkitkan, gen tersebut akan digabungkan menjadi sebuah individu atau kromosom. Kromosom yang dibangkitkan sesuai dengan jumlah populasi yang diinputkan diawal dan kromosom yang telah diinisialisasi tersebut akan digunakan pada proses selanjutnya.

### 3.2.4.2 Reproduksi

Proses reproduksi merupakan proses untuk menghasilkan keturunan dari individu-individu yang ada di populasi. Keturunan atau anak tersebut didapatkan dari individu induk yang direproduksi untuk menemukan solusi yang setidaknya mendekati optimal. Himpunan keturunan ini ditempatkan dalam penampungan *offspring*. Pada proses reproduksi ini terdapat dua operator yang digunakan, yaitu *crossover* dan mutasi. Proses *crossover* pada penelitian ini menggunakan metode *extended intermediate crossover* yang membentuk *offspring* baru dengan mengkombinasikan nilai gen dari dua individu yang dipilih. Pada proses *crossover,* metode ini dipilih karena mampu memberikan individu baru yang berbeda dengan induknya. Sehingga proses *crossover* nantinya akan memberikan variasi yang beragam pada himpunan solusi dari permasalahan yang menggunakan pengkodean *real* ini. Pada dasarnya setiap satu pasang induk yang dilakukan proses *crossover* akan menghasilkan dua buah anak sebagai hasil *offspring*. Proses crossover akan berhenti ketika *offspring* telah terpenuhi sebanyak *Cr* × ukuran populasi (*Popsize)*.

Metode *extended intermediate crossover* akan menghasilkan *offspring* dari kombinasi nilai dua induk menurut (Muhlenbein & Schlierkamp-Voosen, 1993). Misalkan P1 dan P2 adalah dua induk untuk *crossover* yang telah dipilih secara acak, maka C1 dan C2 bisa dibangkitkan dengan menggunakan Persamaan 3.10 dan Persamaan 3.11.

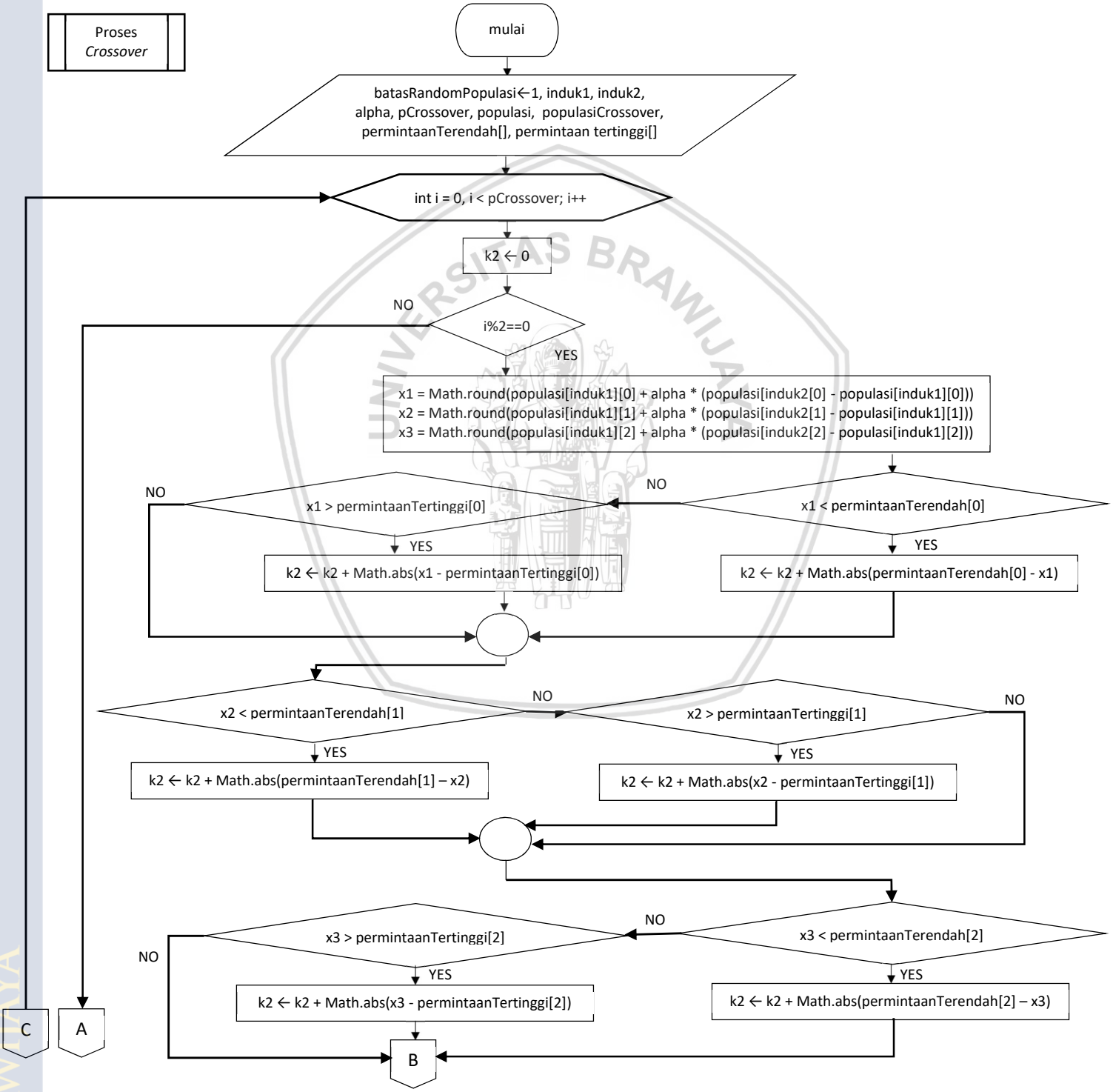
$$C1 = P1 + a\,(P2 - P1) \tag{3.10}$$

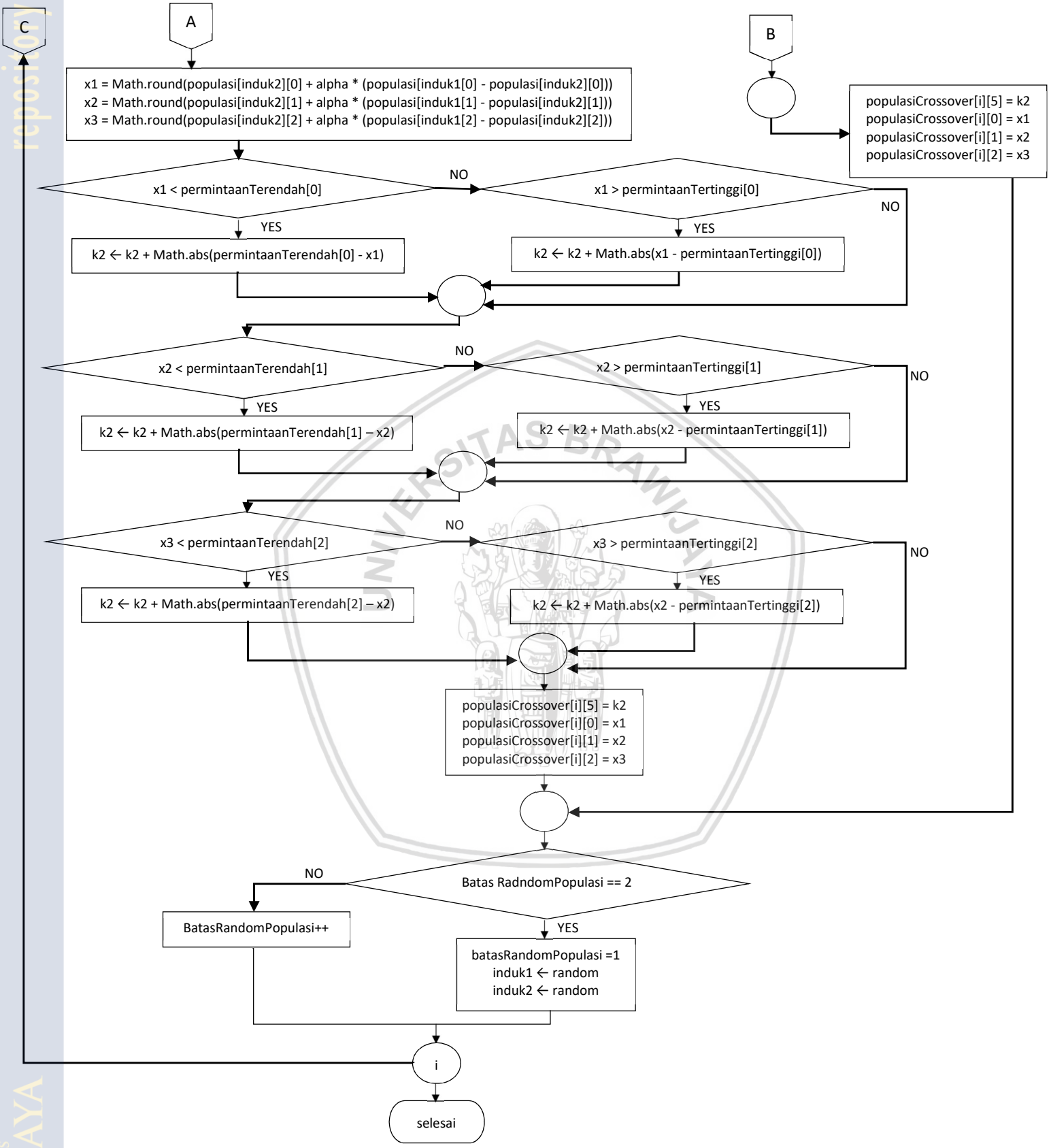
$$C2 = P2 + a\,(P1 - P2) \tag{3.11}$$

Persamaan 3.10 dan Persamaan 3.11 merupakan rumus dari metode *extended intermediate crossover*, adapun keterangan dari persamaan tersebut adalah sebagai berikut:

P1 : induk pertama yang dipilih secara acak dari populasi yang ada.
P2 : induk kedua yang dipilih secara acak dari populasi yang ada.
C1 : *offspring* atau anak pertama yang direproduksi dari induk P1 dan P2.
C2 : *offspring* atau anak kedua yang direproduksi dari induk P1 dan P2.
α : nilai atau bilangan acak yang berada pada interval -0.25 – 1.25

Proses *crossover* yang akan dilakukan pada tahapan ini akan ditunjukkan pada Gambar 3.4.

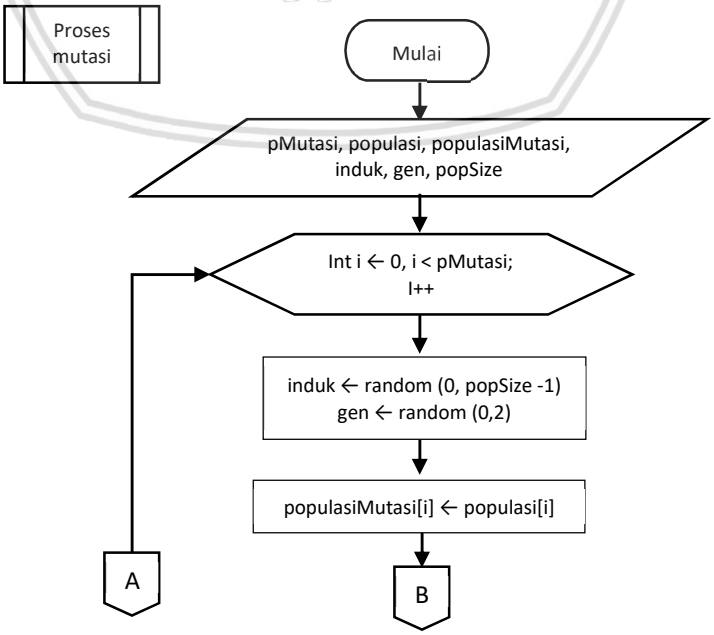


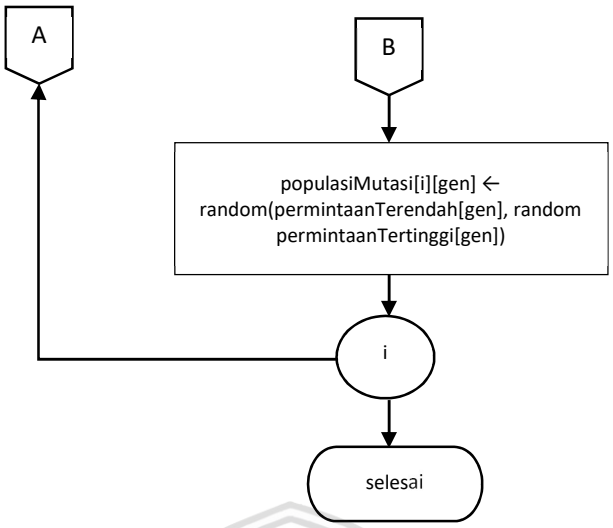



**Gambar 3.4 Diagram Alir Proses Reproduksi *Crossover***

Berdasarkan Gambar 3.4 diagram alir proses reproduksi *crossover* dimulai denngan mengambil kromosom yang telah diinisialisasi pada populasi awal, kromosom tersebut akan direproduksi dengan *extended intermediate crossover*. Individu yang akan dilakukan proses reproduksi *crossover* ini sebanyak *offspring*= *cr*×*popsize*, sehingga proses *crossover* akan berhenti sebanyak nilai *offspring* yang dihasilkan. Lalu, dua individu akan dipilih secara acak untuk dijadikan sebagai *parent*. Dimulai pada *offspring* 1, *parent* yang terpilih tersebut akan dilakukan proses *extended intermediate crossover* dengan menggunakan Persamaan 3.10. Setelah proses tersebut, nilai *child* hasil *crossover* akan mengalami perubahan tipe nilai menjadi *double*, maka hasil tersebut akan diubah kembali ke dalam nilai *integer* dengan melakukan pembulatan nilai. Selanjutnya akan dilakukan langsung penghitungan nilai kendala 2 dari *child* yang telah direproduksi dari *crossover* ini dengan menggunakan Persamaan 3.3, Persamaan 3.4 dan Persamaan 3.5. Setelah *offspring* ke-1 selesai, maka akan dilanjutkan dengan *offspring* ke-2 menggunakan Persamaan 3.11 *extended intermediate crossover* untuk mendapatkan *child* yang kedua. Setelah itu, akan dilakukan cara yang sama dengan proses *crossover* pada *offspring* ke-1 dengan melakukan pembulatan dan penghitungan nilai kendala 2. Proses *crossover* ini akan berhenti apabila jumlah *offspring* sudah terpenuhi.

Setelah melakukan proses *crossover*, pada tahap reproduksi ini juga dilakukan proses mutasi. Mutasi juga merupakan proses untuk menghasilkan keturunan dari individu-individu yang ada didalam populasi. Proses mutasi yang dilakukan adalah mengubah kembali nilai pada gen yang dipilih secara acak, namun pada penelitian ini digunakan rentang permintaan terendah dan permintaan tertinggi sebagai batas pembangkitan kembali nilai gen tersebut. Batas pembangkitan tersebut digunakan untuk menjaga agar hasil soluasi nantinya tetap berada sesuai dengan permintaan pasar yang ada, sehingga diharapkan hasil solusi optimal berdasarkan permintaan yang ada pada bulan sebelumnya. Untuk lebih jelasnya proses mutasi ditunjukkan pada Gambar 3.5.



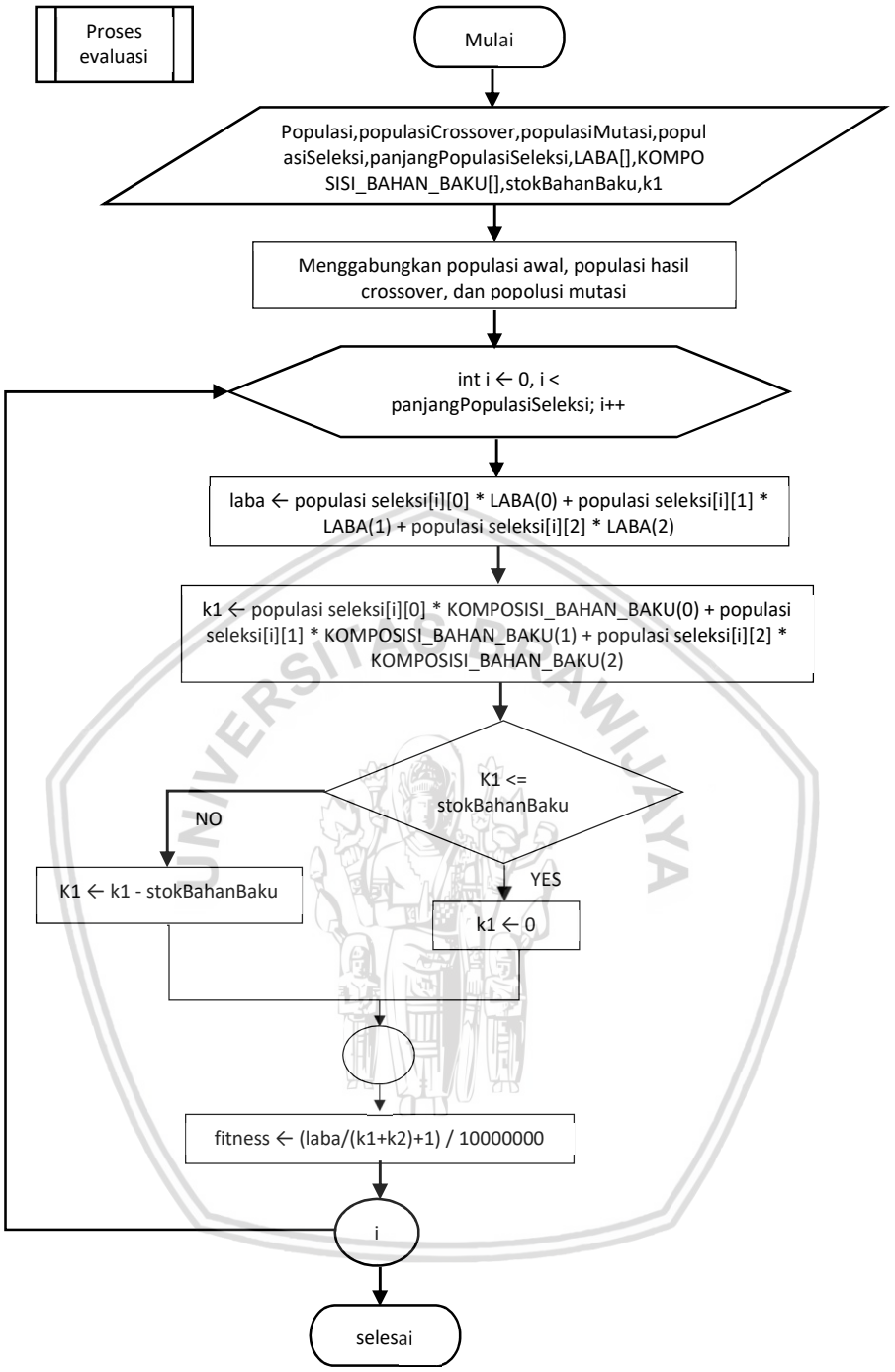



**Gambar 3.5 Diagram Alir Proses Reproduksi Mutasi**

Berdasarkan Gambar 3.5 diagram alir proses reproduksi mutasi dimulai dengan menentukan *offpsring* mutasi sebanyak *offspring=mr×popsize*. Individu yang dilakukan proses mutasi akan dipilih secara acak untuk dijadikan sebagai *parent.* Setelah individu dipilih secara acak, akan ditentukan juga secara acak gen ke berapa yang akan dilakukan proses mutasi. Setelah didapatkan individu serta gen ke berapa nya, maka nilai gen tersebut akan dibangkitkan ulang secara acak dalam rentang permintaan terendah dan permintaan tertinggi. Lalu, proses mutasi akan berhenti setelah jumlah *offspring* sudah terpenuhi.

### 3.2.4.3 Evaluasi

Proses evaluasi merupakan proses untuk menghitung kebugaran (*fitness*) setiap kromosom. Kromosom yang ada pada proses evaluasi merupakan kromosom gabuungan dari populasi awal dan hasil *offspring* yang telah dilakukan dari proses reproduksi baik reproduksi *crossover* maupun reproduksi mutasi. Setelah digabungkan, keseluruhan individu gabungan yang ada akan dihitung untuk didapatkan nilai *fitness* nya. Untuk mendapatkan nilai *fitness* setiap individu menggunakan persamaan perhitungan *fitness* pada subbab sebelumnya. Individu dievaluasi berdasarkan nilai *fitness* nya, semakin besar nilai *fitness* maka akan semakin baik pula solusi yang akan diberikan. Dari hasil proses evaluasi inilah nantinya akan digunakan pada proses seleksi. Untuk proses evaluasi lebih jelasnya ditunjukkan pada Gambar 3.6.



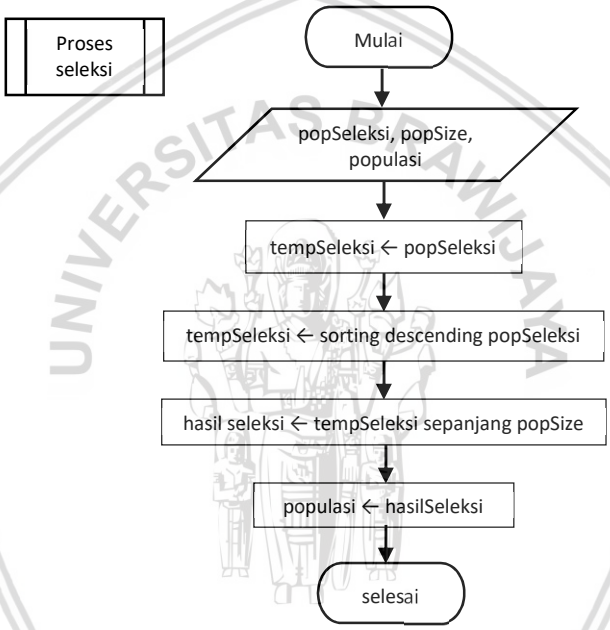

**Gambar 3.6 Diagram Alir Proses Evaluasi**

Berdasarkan Gambar 3.6 diagram alir proses evaluasi merupakan proses penggabungan seluruh individu yang ada, baik dari individu awal hingga individu hasil *crossover* dan individu hasil mutasi. Setelah seluruh individu digabungkan, setiap individu akan dihitung nilai *fitness* nya. Untuk perhitungan *fitness* ini dimulai dengan perhitungan jumlah laba yang didapatkan dari individu solusi yang ada, setelah laba didapatkan akan dihitung nilai kendalanya. Untuk nilai kendala, apabila himpunan solusi yang ada menggunakan bahan baku lebih dari stok maka bahan baku yang berlebih tersebut akan masuk kedalam nilai kendala. Dan

apabila, himpunan solusi tidak menggunakan bahan baku yang berlebih dari stok maka nilai kendala nya akan disimpan sebagai 0. Setelah itu barulah dilakukan perhitungan *fitness* dengan menggunakan Persamaan 3.9.

### 3.2.4.4 Seleksi

Proses seleksi merupakan proses untuk memilih individu dari himpunan populasi dan *offspring* yang dipertahankan hidup pada generasi berikutnya. Semakin besar nilai *fitness* sebuah kromosom maka semakin besar peluangnya untuk terpilih. Hal ini dilakukan agar terbentuk generasi berikutnya yang lebih baik dari generasi sekarang. Pada proses seleksi ini digunakan metode *elitism*, metode *elitism* merupakan metode yang menyaring individu terbaik berdasarkan *fitness* yang terbaik yang akan dipertahankan untuk generasi selanjutnya. Untuk proses seleksi yang lebih jelasnya akan ditunjukkan pada Gambar 3.7.




**Gambar 3.7 Diagram Alir Proses Seleksi**

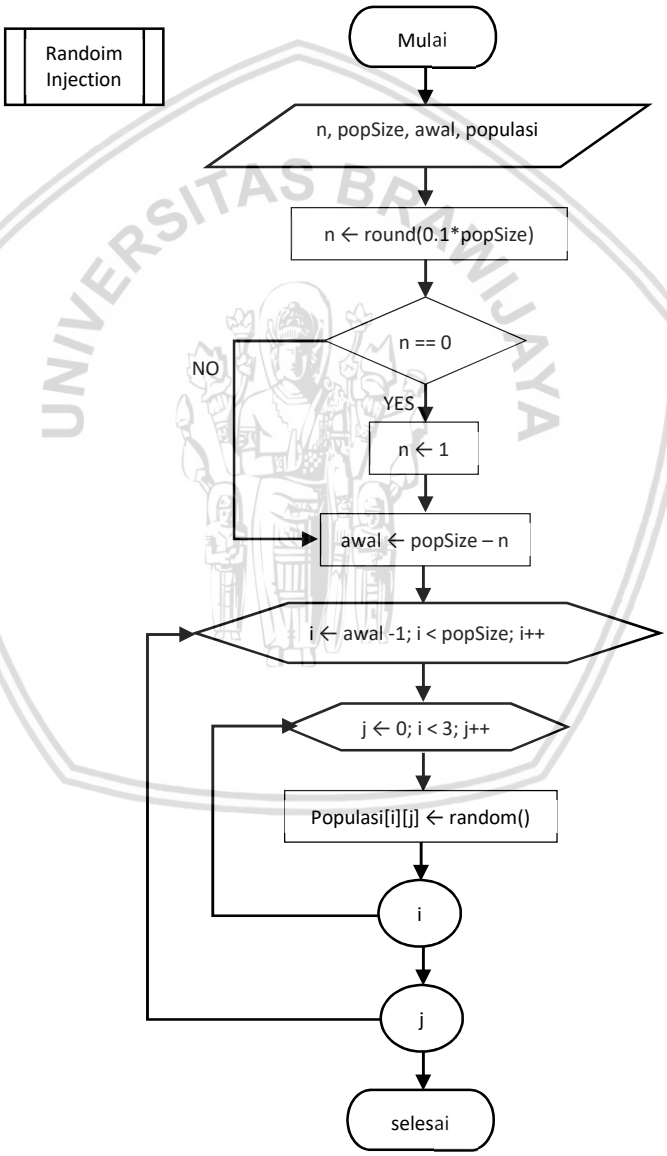
Berdasarkan Gambar 3.7 diagram alir proses seleksi dimulai dengan individu yang telah dievaluasi pada proses sebelumnya akan dikembalikan sejumlah populasi awal. Proses seleksi disini akan mengurutkan indiividu yang ada berdasarkan nilai *fitness* yang terbesar, setelah itu barulah individu teratas akan diambil sebanyak populasi awal untuk digunakan pada populasi awal generasi selanjutnya atau sebagai himpunan solusi jika kondisi berhenti terpenuhi.

### 3.2.4.5 Penanganan Konvergensi Dini

Konvergensi dini merupakan kondisi disaat himpunan solusi mulai tidak berubah atau konvergen mulai diawal iterasi atau generasi, namun himpunan solusi yang diberikan belum optimal. Hal itu bisa disebabkan oleh individu yang tidak bervariasi, sehingga individu hasil reproduksi baik *crossover* maupun mutasi tidak menghasilkan individu yang lebih baik dari sebelumnya. Oleh karena itu, diperlukan penanganan konvergensi dini pada permasalahan tersebut. Terdapat

banyak metode untuk menangani konvergensi dini ini, pada penelitian ini penanganan konvergensi dini ini menggunakan metode *random injection*.

*Random injection* merupakan pembangkitan ulang nilai gen secara random pada *n* individu terakhir, jadi *random injection* melakukan proses seleksi yang hanya memilih sebanyak *popSize* – *n* individu. Nilai *n* individu terakhir yang akan dibangkitkan ulang bisa diatur seberapa besar yang akan dilakukan *random injection*, biasanya 10% dari *popSize* yang akan dilakukan *random injection* sudah cukup untuk menjaga keragaman individu yang ada. Oleh karena itu, *n* individu yang terpilih untuk dibangkitkan ulang bisa diperolah dengan $n$=0.1×*popSize*. Untuk lebih jelasnya proses *random injection* pada penelitian ini dijelaskan pada Gambar 3.8.




**Gambar 3.8 Diagram Alir Proses *Random Injection***

Berdasarkan Gambar 3.8 diagram alir proses *random injection* terlihat bahwa *n* individu yang akan dibangkitkan ulang sebanyak 10% dari *popSize*. *Random*

*injection* ini tentunya tidak akan dilakukan pada setiap iterasi, hal itu dikarenakan apabila dilakukan terus menerus setiap iterasi tentunya akan memakan waktu komputasi yang lebih lama. Pada permasalahan ini, *random injection* dilakukan setiap 10 iterasi hingga ditemukannya iterasi maksimal. Jadi, pada setiap iterasi ke 10 dan kelipatannya, akan dilakukan *random injection* pembangkitan ulang nilai gen secara *random* sebanyak *n* individu terakhir.

### 3.2.5 Perhitungan Manual

Untuk menyelesaikan optimasi jumlah produksi metal roof, langkah awal yang harus dilakukan adalah menentukan parameter input yang akan digunakan pada proses optimasi. Adapun parameter input yang digunakan adalah *crossover rate(Cr)*, *mutation rate(Mr)*, ukuran populasi, maksimal generasi, serta permintaan tertinggi dan permintaan terendah tiap item. Misalkan parameter input ditetapkan sebagai berikut :

- *Crossove Rate(Cr)* = 0.4
- *Mutation Rate(Mr)* = 0.6
- Ukuran populasi = 5
- Maksimal generasi = 1
- Stok bahan baku(kg) = 7282

Dalam parameter input juga terdapat permintaan terendah dan permintaan tertinggi sebagai rentang batas pembangkitan gen dalam individu/kromosom yang akan digunakan dalam proses optimasi. Rentang batas permintaan terendah dan tertinggi ini merupakan batas agar jumlah produksi nanti nya sesuai dengan permintaan pasar dan penggunaan bahan baku tersebut dapat lebih optimal dan sesuai dengan permintaan sebelumnya. Adapun batas permintaan terendah dan permintaan tertinggi yang digunakan ditunjukkan pada Tabel 3.3.

**Tabel 3.3 Batas Pembangkitan Nilai Gen**

| | **Spandek(x1)** | **Zigzag(x2)** | **Zigzag pasiran(x3)** |
|---|---|---|---|
| **batas minimal pembangkitan nilai gen** | 35 | 10 | 1 |
| **batas maksimal pembangkitan nilai gen** | 510 | 6000 | 9000 |

Setelah ditentukan masukan yang akan digunakan, selanjutnya dilakukan pembangkitan nilai gen dalam batas minimal dan batas maksimal sesuai dengan Tabel 3.3. Karena ukuran populasi telah ditentukan yang digunakan adalah 5, maka individu yang akan dibangkitkan adalah sebanyak 5 individu. 5 individu tersebut dibangkitkan secara acak seperti yang ditunjukkan pada Tabel 3.4.

**Tabel 3.4 Inisialisasi populasi awal**

| *Parent* | x1 | x2 | x3 |
|---|---|---|---|
| p1 | 303 | 501 | 4242 |
| p2 | 240 | 5147 | 107 |
| p3 | 485 | 4006 | 7415 |
| p4 | 208 | 1812 | 5411 |
| p5 | 241 | 1138 | 8428 |

Pada Tabel 3.4 p1 adalah individu/kromosom yang menjadi himpunan solusi pada proses ini. Setelah nilai *random* pada individu-individu telah dibangkitkan sebanyak populasi yang ditentukan, maka selanjutnya akan dilakukan perhitungan nilai *fitness*. Perhitungan nilai *fitness* ini akan dilakukan sesuai dengan rumus perhitungan *fitness* yang telah dijelaskan pada Subbab 3.2.3. Langkah-langkah yang harus dilakukan untuk mendapatkan nilai *fitness* adalah sebagai berikut:

1. Hitung nilai penalti

   Berdasarkan inisialisasi awal pada Tabel 3.4 p1 akan digunakan sebagai individu untuk contoh perhitungan nilai *fitness*. Nilai gen pada individu p1 adalah [303, 501, 4242] , lalu hitung nilai penalti menggunakan Persamaan 3.2. Nilai penalti ini didapatkan dengan menghitung kendala1 dan kendala2. Kendala 1 didapatkan dengan menggunakan nilai kebenaran dari 1.45*x1+1.03*x2+1.03*x3 ≤ stok bahan baku. Jika kondisi kebenaran bernilai *true*, maka nilai penalti akan bernilai 0. Dan jika kondisi bernilai *false*, maka nilai penalti didapatkan dari 1.45*x1+1.03*x2+1.03*x3 – stok bahan baku. Pada contoh individu p1, perhitungan nilai kendala1 nya adalah sebagai berikut:

   $$kendala1 = \ 1.45 * 303 \ + \ 1.03 * 501 \ + \ 1.03 * 4242$$

   $$kendala1 = 5324.64$$

   Hasil dari perhitungan tersebut akan dicek kondisi kebenarannya, 5324.64 ≤ 7282. Kondisi kebenaran pada perhitungan p1 bernilai *true*, maka didapatkan nilai kendala k1 adalah 0. Selanjutnya menghitung nilai kendala2, diketahui nilai p1 adalah [303, 501, 5242]. Dimulai dengan gen pertama, nilai gen tersebut adalah 303. Nilai gen pertama ini dicek menggunakan Persamaan 3.3 untuk mendapatkan nilai selisih1. Nilai permintaan terendah gen pertama adalah 35 dan nilai permintaan tertinggi gen pertama adalah 510. Maka gen pertama ini berada didalam rentang batas permintaan terendah dan permintaan tertinggi, sehingga nilai selisih1 nya ada 0. Apabila gen pertama ini terletak dibawah permintaan terendah, maka nilai selisih 1 akan dicari dengan nilai batas permintaan terendah dikurangi gen pertama. Sebaliknya, apabila berada diatas permintaan tertinggi, maka nilai gen tersebut dikurangi dengan batas permintaan tertinggi untuk didapatkan nilai selisihnya.

Lalu gen kedua bernilai 501, maka nilai tersebut dicek tergolong pada keadaan yang mana menggunakan Persamaan 3.4. Nilai permintaan terendah gen kedua adalah 10, dan nilai permintaan tertinggi gen kedua adalah 6000. Maka gen pertama ini berada didalam rentang batas permintaan terendah dan permintaan tertinggi, sehingga nilai selisih2 nya adalah 0. Untuk gen ketiga juga menggunakan proses yang sama seperti gen pertama dan gen kedua. Untuk nilai selisi 3 nya adalah 0, karena nilai gen ketiga ini adalah 4242. Sedangkan, nilai batas permintaan terendah nya adalah 1 dan nilai batas permintaan tertinggi adalah 9000. Sehingga, gen ketiga ini berada dalam rentang batas permintaan terendah dan permintaan tertinggi. Setelah didapatkan nilai selisih tiap gen, dilanjutkan dengan menjumlahkan seluruh selisih tersebut dan disimpan dalam nilai kendala 2 seperti pada Persamaan 3.6.

$$kendala2 = 0 + 0 + 0$$

$$kendala2 = 0$$

Setelah didapatkan nilai kendala-kendala tersebut, maka akan dilanjutkan untuk mendapatkan nilai penalti dari kromosom tersebut dengan menggunakan Persamaan 3.1.

$$pinalti = kendala1 + kendala2$$

$$pinalti = 0 + 0$$

$$pinalti = 0$$

Didapatkan nilai penalti nya adalah 0, berarti kromosom tersebut tidak ada melanggar kendala-kendala yang ada.

2. Hitung total laba

Nilai total laba didapatkan menggunakan Persamaan 3.7. nilai total laba ini didapatkan dari perhitungan jumlah produksi yang menjadi himpunan solusi dikalikan dengan laba per itemnya. Sehingga untuk individu p1, perhitungan total labanya adalah sebagai berikut:

$$total\ laba = 4500 * \mathrm{x1} + 4750 * \mathrm{x2} + 37500 * \mathrm{x3}$$

$$total\ laba = 4500 * 303 + 4750 * 501 + 10500 * 4242$$

$$total\ laba = 48284250$$

Dari hasil perhitungan diatas didapatkan nilai total laba sebesar 48284250.

3. Hitung *fitness*

Nilai *fitness* didapatkan menggunakan Persamaan 3.9. pada persamaan tersebut terdapat nilai konstanta yang menjadi nilai pembagi agar nilai *fitness* tidak dalam rentang yang terlalu besar. Nilai konstanta yang digunakan dalam proses optimasi ini adalah 10000000, nilai konstanta yang digunakan ini cukup besar dikarenakan nilai total laba yang didapatkan dari perhitungan cukup besar. Oleh karena itu penulis

memilih nilai konstanta 10000000 agar nilai *fitness* tidak dalam rentang yang besar. Adapun untuk individu p1, perhitungan nilai *fitness* nya adalah sebagai berikut:

$$fitness = \frac{total\ laba}{pinalti + 1} / 10000000$$

$$fitness = \frac{48284250}{0 + 1} / 10000000$$

$$fitness = 4.828425$$

Dari hasil perhitungan diatas, didapatkan nilai *fitness* untuk individu p1 sebesar 4.828425.

Dengan menggunakan perhitungan nilai *fitness* pada individu p1 diatas, dilakukan proses yang sama untuk mendapatkan nilai *fitness* p2, p3, p4 dan p5. Sehingga didapatkan nilai *fitness* populasi awal keseluruhan seperti yang ditunjukkan pada Tabel 3.5.

**Tabel 3.5 Representasi kromosom beserta *fitness***

| Individu | x1 | x2 | x3 | Total Laba | Penalti | | *Fitness* |
|---|---|---|---|---|---|---|---|
| | | | | | k1 | k2 | |
| p1 | 303 | 501 | 4242 | 48284250 | 0 | 0 | 4.828425 |
| p2 | 240 | 5147 | 107 | 26651750 | 0 | 0 | 2.665175 |
| p3 | 485 | 4006 | 7415 | 99068500 | 5184.88 | 0 | 9.906332 |
| p4 | 208 | 1812 | 5411 | 66358500 | 459.29 | 0 | 6.635804 |
| p5 | 241 | 1138 | 8428 | 94984000 | 2920.43 | 0 | 9.498108 |

Setelah didapatkan keseluruhan nilai *fitness* individu-individu yang ada, maka selanjutnya akan dilakukan proses reproduksi yang terdiri dari *crossover* dan mutasi. Proses reproduksi yang pertama dilakukan adalah *crossover*, tahapan yang akan dilakukan ditunjukkan pada Gambar 3.4 sebelumnya pada subbab 3.2.4. Langkah-langkah *crossover* yang dilakukan adalah sebagai berikut:

1. Tentukan jumlah *offspring*

   Jumlah *offspring* ini didapatkan dari nilai *cr* dikalikan dengan ukuran populasi. Nilai *cr* dan ukuran populasi telah ditentukan sebelumnya, yaitu cr = 0.4 dan ukuran populasi = 5. Maka perhitungan jumlah *offspring* adalah sebagai berikut :

   $$offspring = 0.4 * 5$$

   $$offspring = 2$$

   Offspring yang didapatkan adalah 2, yang berarti proses *crossover* nantinya akan memberikan 2 keturunan baru. Setiap *crossover*

menghasilkan 2 keturunan/anak, maka untuk permasalahan ini hanya terdapat 1 kali operasi *crossover*.

2. Pilih dua individu secara acak

Langkah selanjutnya yang dilakukan adalah memilih 2 individu secara acak untuk menjadi *parent* pada operasi *crossover*. Nantinya dari 2 parent ini akan memberikan 2 keturunan baru yang akan disimpan pada *offspring*. Misalkan, berdasarkan pada Tabel 3.4 dari populasi awal yang ada, terpilih indivdu sebagai *parent* untuk proses operasi *crossover* adalah p3 dan p4 seperti pada Tabel 3.6.

**Tabel 3.6 Parent untuk *Crossover***

| ***Parent*** | **x1** | **x2** | **x3** |
|---|---|---|---|
| p3 | 485 | 4006 | 7415 |
| p4 | 208 | 1812 | 5411 |

3. Bangkitkan nilai α

Nilai α yang akan dibangkitkan sesuai dengan panjang kromosom yang digunakan, misal kromosom yang digunakan adalah 3 maka nilai α yang akan dibangkitkan harus 3 juga. Nilai α yang dibangkitkan harus dalam rentang [-0.25 , 1.25]. Misalkan, setelah dibangkitkan secara acak didapatkan nilai α [1.22 , 0.01, 0.55]. maka nilai α tersebut akan dimasukkan kedalam perhitungan proses *Extended Intermediate Crossover*.

4. Proses *Extended Intermediate Crossover*

Proses *crossover* ini dilakukan pada setiap gen antara *parent* yang terpilih untuk dilakukan *crossover*. Pada pemilihan *parent* secara acak sebelumnya didapatkan p3 dan p4 yang akan dilakukan proses operasi *crossover*. Maka gen pertama pada *offspring* c1 akan didapatkan sebagai berikut:

$$c1x1 = p3 + \alpha * (\mathrm{p4} - \mathrm{p3})$$

$$c1x1 = 485 + 1.22 * (208 - 485)$$

$$c1x1 = 432.37$$

Didapatkan nilai c1x1 = 1296.1 namun pada gen pertama *offspring* c1 ternyata memberikan nilai desimal, sedangkan pada himpunan solusi pada proses optimasi ini menggunakan *real code* bernilai *integer*. Sehingga nilai gen pertama pada *offspring* c1 harus dibulatkan kembali ke nilai *integer* dengan ketentuan apabila < 0,5 maka akan dibulatkan ke bawah dan apabila ≥ 0,5 akan dibulatkan ke atas. Pada perhitungan ini nilai c1x1 = 432.37 , maka nilai c1x1 akan dibulatkan menjadi 432.

Nilai desimal tersebut dikarenakan pada operasi *crossover* ini menggunakan α yang berada dalam rentang[-0.25, 1.25], sehingga harus

dilakukan pembulatan untuk mengembalikan himpunan solusi ke dalam nilai *integer*.

Setelah itu, dilakukan cara yang sama untuk mendapatkan nilai gen kedua dan ketiga *offspring* c1. Sehingga hasil *offspring* c1 adalah [432, 1680, 7655]. Lalu untuk mendapatkan *offspring* c2 menggunakan Persamaan 4.11 dan dilakukan proses yang sama seperti mendapatkan nilai c1. Karena pada permasalahan ini operasi *crossover* hanya mencari 2 *offspring*, maka proses *crossover* akan berhenti setelah 2 *offspring* didapatkan. Apabila *offspring* yang harus dilakukan adalah 3, maka proses *crossover* harus menentukan 2 individu baru lagi untuk melakukan proses operasi *crossver* selanjutnya. Hasil offspring proses *crossover* ditunjukkan pada Tabel 3.7.

**Tabel 3.7 *Offspring* Hasil *Crossover***

| ***Parent*** | ***Offspring*** | **x1** | **x2** | **x3** |
|---|---|---|---|---|
| p3>< p4 | c1 | 432 | 1680 | 7655 |
| | c2 | 261 | 4138 | 5171 |

Selanjutnya setelah proses reproduksi *crossover* selesai, dilanjutkan dengan proses reproduksi kedua yaitu mutasi. Tahapan yang akan dilakukan untuk proses mutasi ditunjukkan pada Gambar 3.5. Langkah-langkah mutasi yang dilakukan adalah sebagai berikut:

1. Tentukan jumlah *offspring*

   Jumlah *offspring* ini didapatkan dari nilai *mr* dikalikan dengan ukuran populasi. Nilai *mr* dan ukuran populasi telah ditentukan sebelumnya, yaitu mr = 0.6 dan ukuran populasi = 5. Maka perhitungan jumlah *offspring* adalah sebagai berikut :

   $$offspring = 0.6 * 5$$

   $$offspring = 3$$

   *Offspring* yang didapatkan adalah 3, yang berarti proses mutasi antinya akan memberikan 3 keturunan baru.

2. Tentukan individu secara acak

   Pada tahapan ini yang dilakukan adalah memilih 1 individu secara acak untuk dijadikan sebagai *parent*. Misalkan, berdasarkan pada Tabel 3.4 terpilih p2 sebagai *parent* untuk proses mutasi yang ditunjukkan pada Tabel 3.8.

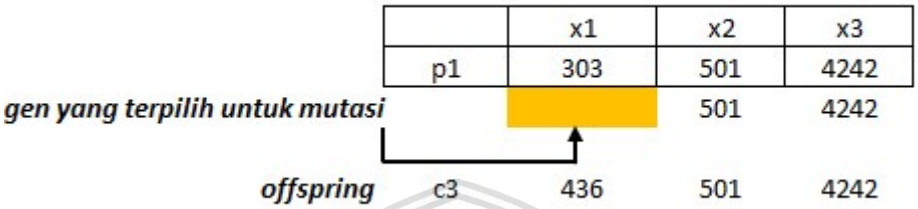
**Tabel 3.8 *Parent* Proses Mutasi**

| ***Parent*** | **x1** | **x2** | **x3** |
|---|---|---|---|
| p1 | 303 | 501 | 4242 |

3. Bangkitkan ulang nilai yang terpilih

   Pada bagian ini, nilai pada individu p2 yang sudah terpilih secara acak nantinya akan dibangkitkan ulang sebagai proses mutasi. Mutasi pada

individu p2 dimulai dengan memilih secara acak gen yang akan dibangkitkan ulang antara x1, x2 atau x3. Misalkan gen yang terpilih adalah gen x1, maka nilai pada gen x1 dibangkitkan ulang secara acak dalam batas rentang permintaan terendah dan permintaan tertinggi. x1 merupakan item spandek, maka rentang permintaan terendah dan permintaan tertinggi yang digunakan adalah rentang pada item spandek yaitu 35 hingga 510. Proses mutasi ini dijelaskan seperti pada Gambar 3.9.




**Gambar 3.9 Proses Mutasi**

Setelah didapatkan *offspring* c3 [436, 501, 4242], langkah ini terus diulangi sampai jumlah *offspring* = *mr**ukuran populasi terpenuhi. Karena sebelumnya jumlah *offspring* telah didapatkan 3 *offspring*, maka proses yang sama akan terus diulangi sampai memenuhi 3 *offspring*. Individu hasil *offspring* mutasi yang didapatkan pada masalah ini ditunjukkan pada Tabel 3.9.

**Tabel 3.9 *Offspring* Hasil Mutasi**

| *Parent* | *Offspring* | x1 | x2 | x3 |
|---|---|---|---|---|
| p1 | c3 | 436 | 501 | 4242 |
| p2 | c4 | 240 | 5147 | 2013 |
| p5 | c5 | 326 | 1138 | 8428 |

Setelah proses reproduksi operasi *crossover* dan mutasi selesai, proses selanjutnya yang harus dilakukan adalah evaluasi. Evaluasi ini adalah penggabungan seluruh individu dari populasi awal hingga *offspring* hasil *crossover* dan *offspring* hasil mutasi. Lalu, setelah itu dihitung nilai *fitness* dari *offspring* hasil *crossover* dan *offspring* hasil mutasi. Seluruh penggabungan beserta nilai *fitness* ditunjukkan pada Tabel 3.10.

**Tabel 3.10 Proses Evaluasi**

| Individu | x1 | x2 | x3 | Total laba | Penalti | | *Fitness* |
|---|---|---|---|---|---|---|---|
| | | | | | k1 | k2 | |
| p1 | 303 | 501 | 4242 | 48284250 | 0 | 0 | 4.828425 |
| p2 | 240 | 5147 | 107 | 26651750 | 0 | 0 | 2.665175 |
| p3 | 485 | 4006 | 7415 | 99068500 | 5184.88 | 0 | 0.00191 |
| p4 | 208 | 1812 | 5411 | 66358500 | 459.29 | 0 | 0.014417 |
| p5 | 241 | 1138 | 8428 | 94984000 | 2920.43 | 0 | 0.003251 |
| c1 | 432 | 1680 | 7655 | 90309915 | 2960.852 | 0 | 0.003049 |
| c2 | 261 | 4138 | 5171 | 75117085 | 2683.318 | 0 | 0.002798 |
| c3 | 436 | 501 | 4242 | 48882750 | 0 | 0 | 4.888275 |
| c4 | 240 | 5147 | 2013 | 46664750 | 440.8 | 0 | 0.010562 |
| c5 | 326 | 1138 | 8428 | 95366500 | 3043.68 | 0 | 0.003132 |

Setelah proses evaluasi penggabungan keseleruhan individu selesai, selanjutnya adalah proses seleksi. Berdasarkan hasil evaluasi pada Tabel 3.10 proses seleksi dilakukan dengan pengurutan dari nilai *fitness* yang terbesar hingga yang terkecil seperti yang ditunjukkan pada Tabel 3.11.

**Tabel 3.11 Pengurutan Nilai *Fitness***

| Parent | x1 | x2 | x3 | Total laba | Penalti | | *Fitness* |
|---|---|---|---|---|---|---|---|
| | | | | | k1 | k2 | |
| c3 | 436 | 501 | 4242 | 48882750 | 0 | 0 | 4.888275 |
| p1 | 303 | 501 | 4242 | 48284250 | 0 | 0 | 4.828425 |
| p2 | 240 | 5147 | 107 | 26651750 | 0 | 0 | 2.665175 |
| p4 | 208 | 1812 | 5411 | 66358500 | 459.29 | 0 | 0.014417 |
| c4 | 240 | 5147 | 2013 | 46664750 | 440.8 | 0 | 0.010562 |
| p5 | 241 | 1138 | 8428 | 94984000 | 2920.43 | 0 | 0.003251 |
| c5 | 326 | 1138 | 8428 | 95366500 | 3043.68 | 0 | 0.003132 |
| c1 | 432 | 1680 | 7655 | 90309915 | 2960.852 | 0 | 0.003049 |
| c2 | 261 | 4138 | 5171 | 75117085 | 2683.318 | 0 | 0.002798 |
| p3 | 485 | 4006 | 7415 | 99068500 | 5184.88 | 0 | 0.00191 |

Dari hasil yang telah diurutkan berdasarkan *fitness* yang tertinggi seperti Tabel 3.11, selanjutnya akan diambil individu yang terpilih sesuai dengan ukuran populasi yang telah diinputkan diawal. Sebelumnya ukuran populasi telah

ditentukan sebanyak 5, maka individu yang dipilih pada permasalahan ini adalah sebanyak 5 individu. Proses pemilihan 5 individu terbaik ini digunakan untuk individu baru pada generasi selanjutnya. Individu yang terpilih untuk generasi selanjutnya ditunjukkan pada gambar 3.12.

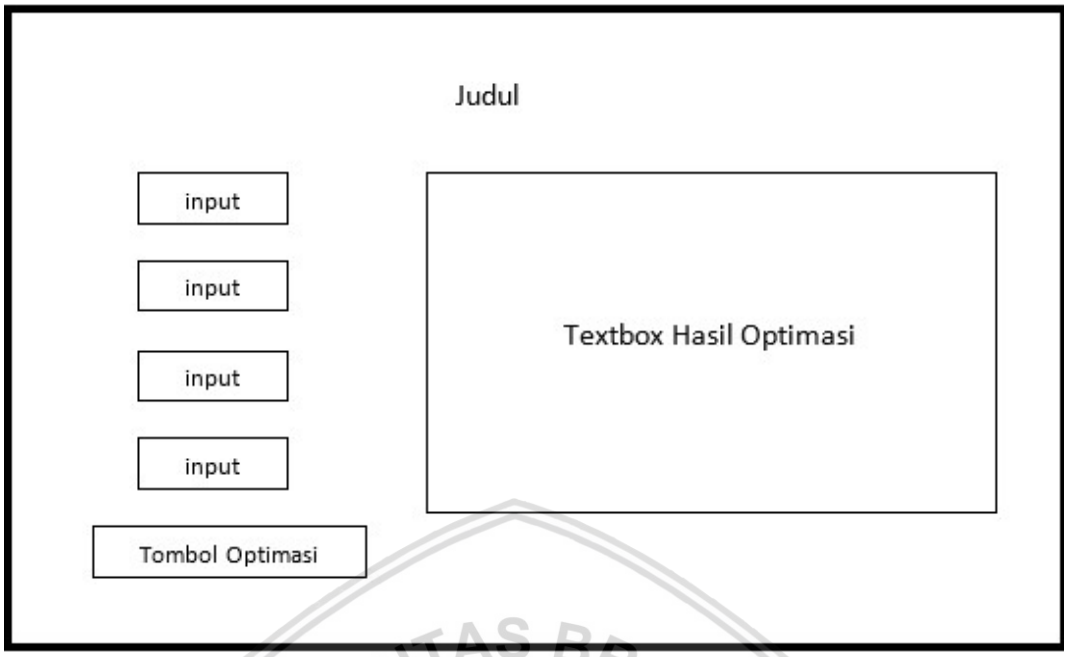
**Tabel 3.12 Individu Hasil Seleksi**

| Individu | | x1 | x2 | x3 | Total laba | Penalti | | *Fitness* |
|---|---|---|---|---|---|---|---|---|
| | | | | | | k1 | k2 | |
| p1 | c3 | 436 | 501 | 4242 | 48882750 | 0 | 0 | 4.888275 |
| p2 | p1 | 303 | 501 | 4242 | 48284250 | 0 | 0 | 4.828425 |
| p3 | p2 | 240 | 5147 | 107 | 26651750 | 0 | 0 | 2.665175 |
| p4 | p4 | 208 | 1812 | 5411 | 66358500 | 459.29 | 0 | 0.014417 |
| p5 | c4 | 240 | 5147 | 2013 | 46664750 | 440.8 | 0 | 0.010562 |

Apabila tercapai kondisi berhenti (*stopping condition*) maka himpunan solusi yang akan diambil adalah individu yang terbaik pada hasil seleksi akhir. Misal ini adalah kondisi berhenti, maka p1 akan menjadi individu solusi dari permasalahan ini. Kombinasi jumlah produksi yang dihasilkan adalah 436 meter spandek, 501 lembar zigzag, dan 4242 lembar zigzag dengan nilai *fitness* 4.888275 serta total keuntungan Rp. 48.882.750.

### 3.2.6 Perancangan Antarmuka

Pada perancangan antarmuka ini, hanya terdapat 1 tampilan utama yang akan menampilkan tampilan keluaran dari proses optimasi menggunakan Algoritme Genetika ini. Pada tampilan utama ini terdapat judul dari penelitian yang dilakukan, disertai dengan inputan yang digunakan untuk mendapatkan hasil optimasi. Terdapat juga tombol optimasi untuk menjalankan proses, lalu hasil dari proses tersebut akan ditampilkan pada *textbox* hasil optimasi. Perancangan antarmuka dari penelitian ini ditampilkan seperti pada Gambar 3.10.




**Gambar 3.10 Perancangan Antarmuka**

## 3.2.7 Perancangan Pengujian

Pada Subbab ini peneliti akan membahas tentang perancangan pada pengujian data terhadap implementasi dari proses yang telah dilakukan. Untuk pengujian masing-masing kondisi akan dijalankan sebanyak 5 kali proses percobaan, kemudian diambil hasil rata-rata yang digunakan untuk memaksimalkan hasil karena pada inisialisasi bobot pada Algoritme Genetika digunakan nilai acak. Dalam penelitian ini dilakukan tiga macam pengujian yaitu Pengujian Ukuran Populasi, Pengujian Kombinasi *cr* dan *mr*, dan Pengujian Jumlah Generasi. Hasil proses pengujian dalam penelitian ini merupakan hasil optimasi sistem dalam kasus jumlah produksi metal roof untuk mendapatkan jumlah produksi yang optimal.

### 3.2.7.1 Perancangan Pengujian Ukuran Populasi

Dalam perancangan pengujian ukuran populasi ini nantinya pada pengujian akan dilakukan sebanyak 5 pengujian setiap ukuran populasi yang digunakan, lalu akan diambil nilai rata-ratanya untuk diambil satu nilai tertinggi sebagai ukuran populasi terbaik. Ukuran populasi yang digunakan adalah kelipatan 10 dimulai dari 10 sampai dengan 100. Untuk parameter lainya ditetapkan antara lain nilai *cr* = 0.5, *mr* = 0.5, dan maksimal generasi = 1000. Berikut adalah tabel perancangan pengujian ukuran populasi yang ditunjukkan pada Tabel 3.13.

**Tabel 3.13 Perancangan Pengujian Ukuran Populasi**

| Ukuran Populasi | Percobaan ke - | | | | | Rata-rata |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| 10 | | | | | | |
| 20 | | | | | | |
| 30 | | | | | | |
| 40 | | | | | | |
| 50 | | | | | | |
| 60 | | | | | | |
| 70 | | | | | | |
| 80 | | | | | | |
| 90 | | | | | | |
| 100 | | | | | | |

### 3.2.7.2 Perancangan Pengujian Kombinasi *cr* dan *mr*

Dalam perancangan pengujian kombinasi *cr* dan *mr* ini nantinya pada pengujian akan dilakukan sebanyak 5 pengujian setiap kombinasi *cr* dan *mr* yang digunakan, lalu akan diambil nilai rata-ratanya untuk diambil satu nilai tertinggi sebagai kombinasi *cr* dan *mr* terbaik. Kombinasi *cr* dan *mr* yang digunakan pada pengujian ini merupakan kelipatan 0,1 antara 0 dan 1. Untuk nilai *cr* dimulai dari 0.9-0.1, dan nilai *mr* dimulai dari 0.1-0.9. Untuk parameter ukuran populasi terbaik didapatkan dari pengujian ukuran populasi, dan maksimal generasi = 1000. Berikut adalah tabel perancangan pengujian kombinasi *cr* dan *dr* yang ditunjukkan pada Tabel 3.14.

**Tabel 3.14 Peracangan Pengujian Kombinasi *cr* dan *mr***

| Kombinasi | | Percobaan ke - | | | | | Rata-rata |
|---|---|---|---|---|---|---|---|
| *cr* | *mr* | 1 | 2 | 3 | 4 | 5 | |
| 0.9 | 0.1 | | | | | | |
| 0.8 | 0.2 | | | | | | |
| 0.7 | 0.3 | | | | | | |
| 0.6 | 0.4 | | | | | | |
| 0.5 | 0.5 | | | | | | |
| 0.4 | 0.6 | | | | | | |
| 0.3 | 0.7 | | | | | | |

| Kombinasi | | Percobaan ke - | | | | | Rata-rata |
|---|---|---|---|---|---|---|---|
| *cr* | *mr* | 1 | 2 | 3 | 4 | 5 | |
| 0.2 | 0.8 | | | | | | |
| 0.1 | 0.9 | | | | | | |

### 3.2.7.3 Perancangan Pengujian Jumlah Generasi

Dalam perancangan pengujian jumlah generasi ini nantinya pada pengujian akan dilakukan sebanyak 5 pengujian setiap generasi yang digunakan, lalu akan diambil nilai rata-ratanya untuk diambil satu nilai tertinggi sebagai generasi terbaik. Jumlah generasi yang digunakan pada pengujian ini merupakan kelipatan 25 dimulai dari 25 sampai dengan 250. Untuk parameter ukuran populasi, digunakan nilai ukuran populasi yang didapatkan dari pengujian ukuran populasi. Untuk parameter kombinasi *cr* dan *mr*, digunakan nilai kombinasi *cr* dan *mr* yang didapatkan dari pengujian kombinasi *cr* dan *mr*. Berikut adalah tabel perancangan pengujian generasi ditunjukkan pada Tabel 3.15.

**Tabel 3.15 Perancangan Pengujian Jumlah Generasi**

| Banyak Generasi | Percobaan ke - | | | | | Rata-rata |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| 25 | | | | | | |
| 50 | | | | | | |
| 75 | | | | | | |
| 100 | | | | | | |
| 125 | | | | | | |
| 150 | | | | | | |
| 175 | | | | | | |
| 200 | | | | | | |
| 225 | | | | | | |
| 250 | | | | | | |

# BAB 4 IMPLEMENTASI

Pada bab 4 implementasi ini akan menjelaskan tentang implementasi optimasi jumlah produksi metal roof yang menggunakan Algoritme Genetika, bab ini mengacu pada Subbab perancangan algoritme yang telah dibuat. Implementasi yang akan dijelaskan adalah mengenai spesifikasi perangkat yang digunakan untuk implementasi dan *source code* dari implementasi proses Algoritme Genetika.

## 4.1 Spesifikasi Perangkat

Pada spesifikasi perangkat ini akan menjelaskan tentang spesifikasi dari perangkat yang digunakan pada penelitian ini untuk melakukan implementasi. Spesifikasi perangkat yang digunakan terbagi menjadi 2 yaitu, spesifikasi perangkat keras dan perangkat lunak.

### 4.1.1 Spesifikasi Perangkat Keras

Pada implementasi proses optimasi jumlah produksi metal roof menggunakan Algoritme Genetika ini spesifikasi perangkat keras yang digunakan adalah Laptop Asus X450J dengan spesifikasi sebagai berikut:

1. Processor Intel(R) Core(TM) i7-4700HQ CPU @ 2.40GHz 2.40 GHz
2. Memory (RAM) : 4GB
3. Harddisk : 1TB

### 4.1.2 Spesifikasi Perangkat Lunak

Pada implementasi proses optimasi jumlah produksi metal roof menggunakan Algoritme Genetika ini spesfikasi perangkat lunak yang digunakan adalah sebagai berikut:

1. Sistem Operasi Windows-64bit
2. NetBeans IDE 8.0.1

## 4.2 Implementasi *Source Code*

Implementasi *source code* proses optimasi jumlah produksi metal roof menggunakan Algoritme Genetika disini menggunakan bahasa pemrograman *java*. Pada Subbab ini terdapat beberapa proses yaitu, implementasi inisialisasi kromosom awal, implementasi proses *crossover*, implementasi proses mutasi, implementasi evaluasi dan perhitungan *fitness*, serta implementasi proses seleksi.

### 4.2.1 Implementasi Inisialisasi Kromosom Awal

Inisialisasi kromosom awal merupakan langkah awal dalam Algoritme Genetika. Langkah awal dilakukan dengan pembangkitan tiap gen sesuai dengan batas permintaan terendah dan permintaan terendah, yang dimana gabungan gen tersebut nantinya menjadi kromosom/individu yang akan diproses. Inisialisasi

kromosom awal ini telah dibahas pada Subbab 3.2.4.1 secara detail, proses tersebut telah diimplementasikan pada *Source Code* 4.1.

```
Method inisialisasiKromosom
1  public void inisialisasiKromosom(){
2          for (int i = 0; i < this.p.length; i++) {
3              for (int j = 0; j < 3; j++) {
4                  this.p[i][j] = this.permintaanTerendah[j] +
5  (int)(Math.random() * ((this.permintaanTertinggi[j] -
6  (this.permintaanTerendah[j]) + 1)));
7              }
8          }
9      }
```

***Source Code* 4.1 Inisialisasi Kromosom Awal**

Penjelasan *Source Code* 4.1:

1. Baris 1 merupakan method untuk melakukan proses inisialisasi kromosom awal.
2. Baris 2-9 merupakan proses pembangkitan nilai setiap gennya secara *random* sesuai dengan rentang batas permintaan terendah dan permintaan tertinggi yang dimana kromosomnya sebanyak panjang populasi.

### 4.2.2 Implementasi Proses *Crossover*

Proses *crossover* merupakan salah satu langkah reproduksi dalam Algoritme Genetika, proses ini dilakukan setelah pembangkitan kromosom. Proses crossover ini menggunakan *Extended Intermediate Crossover*, proses *crossover* disini telah dibahas pada Subbab 3.2.4.2 secara detail, proses tersebut telah diimplementasikan pada *Source Code* 4.2.

```
Method crossover
1   public void crossover(){
2           int batasRandomPopulasi = 1;
3           int ind1 = 0 + (int)(Math.random()*(((this.popSize - 1) -
4   (0)+1)));
5           int ind2 = 0 + (int)(Math.random()*(((this.popSize - 1) -
6   (0)+1)));
7
8               double x1, x2, x3;
9               if(i % 2 == 0){
10                  x1 = Math.round(this.p[ind1][0] + this.alfa[0] *
11  (this.p[ind2][0] - this.p[ind1][0]));
12                  x2 = Math.round(this.p[ind1][1] + this.alfa[1] *
13  (this.p[ind2][1] - this.p[ind1][1]));
14                  x3 = Math.round(this.p[ind1][2] + this.alfa[2] *
15  (this.p[ind2][2] - this.p[ind1][2]));
16
17                  if(x1 < this.permintaanTerendah[0]){
18                      k2 += Math.abs(this.permintaanTerendah[0] -
19  x1);
20                  }else if(x1 > this.permintaanTertinggi[0]){
21                      k2 += Math.abs(x1 -
22  this.permintaanTertinggi[0]);
23                  }
24                  if(x2 < this.permintaanTerendah[1]){
25                      k2 += Math.abs(this.permintaanTerendah[1] -
26  x2);
27                  }else if(x2 > this.permintaanTertinggi[1]){
28
```

```
                    k2 += Math.abs(x2 -
this.permintaanTertinggi[1]);
                }
                if(x3 < this.permintaanTerendah[2]){
                    k2 += Math.abs(this.permintaanTerendah[2] -
x3);
                }else if(x3 > this.permintaanTertinggi[2]){
                    k2 += Math.abs(x3 -
this.permintaanTertinggi[2]);
                }
                this.pc[i][5] = k2;

                this.pc[i][0] = x1;
                this.pc[i][1] = x2;
                this.pc[i][2] = x3;
            }else{
                x1 = Math.round(this.p[ind2][0] + this.alfa[0] *
(this.p[ind1][0] - this.p[ind2][0]));
                x2 = Math.round(this.p[ind2][1] + this.alfa[1] *
(this.p[ind1][1] - this.p[ind2][1]));
                x3 = Math.round(this.p[ind2][2] + this.alfa[2] *
(this.p[ind1][2] - this.p[ind2][2]));

                if(x1 < this.permintaanTerendah[0]){
                    k2 += Math.abs(this.permintaanTerendah[0] -
x1);
                }else if(x1 > this.permintaanTertinggi[0]){
                    k2 += Math.abs(x1 -
this.permintaanTertinggi[0]);
                }
                if(x2 < this.permintaanTerendah[1]){
                    k2 += Math.abs(this.permintaanTerendah[1] -
x2);
                }else if(x2 > this.permintaanTertinggi[1]){
                    k2 += Math.abs(x2 -
this.permintaanTertinggi[1]);
                }
                if(x3 < this.permintaanTerendah[2]){
                    k2 += Math.abs(this.permintaanTerendah[2] -
x3);
                }else if(x3 > this.permintaanTertinggi[2]){
                    k2 += Math.abs(x3 -
this.permintaanTertinggi[2]);
                }
                this.pc[i][5] = k2;

                this.pc[i][0] = x1;
                this.pc[i][1] = x2;
                this.pc[i][2] = x3;
            }

            if(batasRandomPopulasi == 2){
                batasRandomPopulasi = 1;
                ind1 = 0 + (int)(Math.random()*(((this.popSize -
1) - (0)+1)));
                ind2 = 0 + (int)(Math.random()*(((this.popSize -
1) - (0)+1)));
            }else{
                batasRandomPopulasi++;
            }
        }
    }
```

***Source Code* 4.2 Proses *Crossover***

Penjelasan *Source Code* 4.2:

1. Baris 1 merupakan method untuk melakukan proses *crossover*.
2. Baris 2-6 merupakan pemilihan kromosom induk secara *random* yang akan digunakan dalam proses *crossover*.
3. Baris 7-78 merupakan proses reproduksi untuk menghasilkan anak menggunakan *Extended Intermediate Crossover*, lalu hasil setiap anak hasil *crossover* akan langsung dicek apakah nilai setiap gennya diluar rentang batas permintaan terendah dan permintaan terendah. Apabila nilai pada gen tersebut ada yang diluar batas akan langsung dihitung nilai kendalanya.
4. Baris 79-90 merupakan kondisi pengecekan apabila *crossover* menghasilkan lebih dari 1 *offspring*, maka akan dipilih induk baru lagi secara *random* untuk dilakukan proses *crossover* pada *offspring* selanjutnya dan dilakukan proses *crossover* dengan cara yang sama pada baris 7-97.

### 4.2.3 Implementasi Proses Mutasi

Proses mutasi merupakan langkah kedua pada reproduksi dalam Algoritme Genetika, proses ini dilakukan dengan membangkitkan ulang salah satu gen pada kromosom sesuai rentang batas permintaan terendah dan permintaan tertinggi. Proses mutasi disini telah dibahas pada Subbab 3.2.4.2 secara detail, proses tersebut telah diimplementasikan pada *Source Code* 4.3.

Method mutasi

```
1   public void mutasi(){
2           for (int i = 0; i < this.pm.length; i++) {
3               int ind = 0 + (int)(Math.random()*(((this.popSize - 1)
4   - (0)+1)));
5               int gen = 0 + (int)(Math.random()*((2 - (0)+1)));
6
7               this.pm[i] = Arrays.copyOfRange(this.p[ind], 0,
8   this.p[ind].length);
9
10              this.pm[i][gen] = this.permintaanTerendah[gen] +
11                          (int)(Math.random() *
12                          ((this.permintaanTertinggi[gen] -
13  (this.permintaanTerendah[gen]) + 1)));
14          }
15      }
```

***Source Code* 4.3 Proses Mutasi**

Penjelasan *Source Code* 4.3:

1. Baris 1 merupakan method untuk melakukan proses mutasi.
2. Baris 2-5 merupakan proses yang dimulai dengan pemilihan kromosom secara *random*, lalu dari kromosom terpilih akan dipilih juga gen nya secara *random* untuk dilakukan proses mutasi.
3. Baris 6-15 merupakan pengambilan nilai dari kromosom terpilih, lalu menggantikan nilai pada gen yang terpilih dengan cara pembangkitan ulang didalam rentang batas permintaan terendah dan permintaan tertinggi.

### 4.2.4 Implementasi Evaluasi dan Perhitungan *Fitness*

Proses evaluasi merupakan penggabungan semua kromosom/individu dari populasi awal, populasi dari proses *crossover* dan populasi dari proses mutasi. Setelah semua digabungkan dilakukan perhitungan *fitness*, yang dimana prosesnya telah dibahas pada Subbab 3.2.4.3 secara detail, proses tersebut telah diimplementasikan pada *Source Code* 4.4.

Method evaluasi

```
1   public void evaluasi(){
2           for (int i = 0; i < this.p.length; i++) {
3               for (int j = 0; j < this.p[i].length; j++) {
4                   this.ps[i][j] = this.p[i][j];
5               }
6           }
7           for (int i = 0; i < this.pc.length; i++) {
8               for (int j = 0; j < this.p[i].length; j++) {
9                   this.ps[i + this.p.length][j] = this.pc[i][j];
10              }
11          }
12          for (int i = 0; i < this.pm.length; i++) {
13              for (int j = 0; j < this.p[i].length; j++) {
14                  this.ps[i + this.p.length + this.pc.length][j] =
15  this.pm[i][j];
16              }
17          }
18
19          for (int i = 0; i < this.ps.length; i++) {
20              int laba = ((int)this.ps[i][0] * this.LABA[0]) +
21                      ((int)this.ps[i][1] * this.LABA[1]) +
22                      ((int)this.ps[i][2] * this.LABA[2]);
23
24              double k1 = this.ps[i][0] *
25  this.KOMPOSISI_BAHAN_BAKU[0] +
26                      this.ps[i][1] * this.KOMPOSISI_BAHAN_BAKU[1] +
27                      this.ps[i][2] * this.KOMPOSISI_BAHAN_BAKU[2];
28              if(k1 <= this.stokBahanBaku){
29                  k1 = 0;
30              }else{
31                  k1 -= this.stokBahanBaku;
32              }
33
34              double fitness = (laba /((k1 +
35  this.ps[i][5])+1))/10000000;
36              this.ps[i][3] = laba;
37              this.ps[i][4] = k1;
38              this.ps[i][6] = fitness;
39          }
40      }
```

***Source Code* 4.4 Proses Evaluasi dan Perhitungan *Fitness***

Penjelasan *Source Code* 4.4:

1. Baris 1 merupakan method untuk melakukan proses evaluasi.
2. Baris 2-17 merupakan penggabungan seluruh individu, baik dari individu awal, individu hasil *crossover* dan individu mutasi.
3. Baris 18-40 merupakan perhitungan nilai *fitness*.

### 4.2.5 Implementasi Proses Seleksi

Proses seleksi merupakan proses akhir dalam perulangan pada Algoritme Genetika. Proses seleksi merupakan proses pengambilan individu terbaik yang prosesnya telah dibahas pada Subbab 3.2.4.4 secara detail, proses tersebut telah diimplementasikan pada *Source Code* 4.5.

```
Method seleksi
1   public void seleksi(){
2           double tempSeleksi[][] = new
3   double[this.ps.length][this.ps[0].length];
4           double hasilSeleksi[][] = new
5   double[this.popSize][this.ps[0].length];
6           System.arraycopy(this.ps, 0, tempSeleksi, 0,
7   this.ps.length);
8
9           tempSeleksi = elitism(tempSeleksi);
10
11          System.arraycopy(tempSeleksi, 0, hasilSeleksi, 0,
12  this.popSize);
13          updatePopulasi(hasilSeleksi);
14      }
15
16      public double[][] elitism(double [][]pSeleksi){
17          double [][]tempP = pSeleksi;
18
19          Arrays.sort(tempP, new java.util.Comparator<double[]>() {
20              public int compare(double[] a, double[] b) {
21                  return Double.compare(b[b.length - 1], a[a.length
22  - 1]);
23              }
24          });
25          return tempP;
26      }
```

***Source Code* 4.5 Proses Seleksi**

Penjelasan *Source Code* 4.5:

1. Baris 1 merupakan method untuk melakukan proses seleksi.
2. Baris 2-26 merupakan proses seleksi untuk mengambil individu terbaik sebanyak populasi awal untuk sebagai hasil solusi dari iterasi ke n.

### 4.2.6 Implementasi Proses *Random Injection*

Proses *random injection* merupakan proses penanganan konvergensi dini. Proses ini merupakan proses pembangkitan ulang nilai gen yang berada pada posisi bawah seleksi untuk menjaga keragaman individu yang ada pada populasi, untuk proses lebih jelas nya telah dibahas pada Subbab 3.2.4.5 secara detail, proses tersebut telah diimplementasikan pada *Source Code* 4.6.

```
Method RandomInjection
1   public void randomInjection(){
2           int n = 0;
3           double temp = Math.round(0.1 * this.popSize);
4           if(temp == 0){
5               temp = 1;
6           }
7           n = (int) temp;
8           int kurang = this.popSize - n;
9           for (int i = kurang; i < this.popSize; i++) {
10              for (int j = 0; j < 3; j++) {
11                  this.p[i][j] = this.permintaanTerendah[j] +
12                          (int)(Math.random() *
13                          ((this.permintaanTertinggi[j] -
14  (this.permintaanTerendah[j]) + 1)));
15              }
16          }
17      }
```

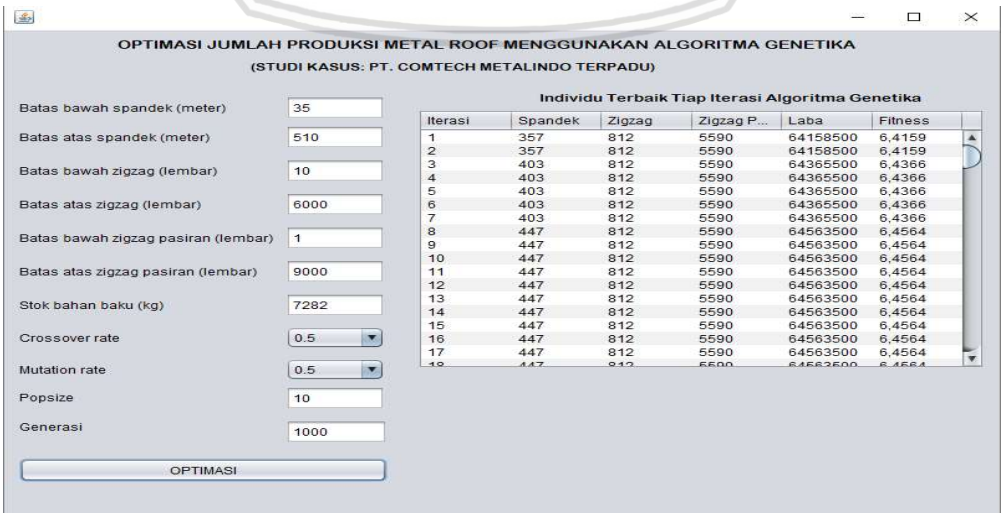
***Source Code* 4.6 Proses *Random Injection***

Penjelasan *Source Code* 4.6:

3. Baris 1 merupakan method untuk melakukan proses *random injection*.
4. Baris 2-17 merupakan proses *random injection* untuk memilih individu terbawah sebanyak *n*=0.1×*popSize*, dan melakukan pembangkitan ulang nilai gen dari individu yang terpilih tersebut didalam rentang batas permintaan terendah dan permintaan tertinggi.

## 4.2.7 Implementasi Antarmuka

Implementasi antarmuka ini merupakan implementasi tampilan utama yang digunakan untuk menunjang hasil optimasi. Pada implementasi ini terdapat judul optimasi dari penelitian ini, pada bagian kiri terdapat juga *textbox* untuk mengisi masukan yang digunakan untuk menyelesaikan permasalahan optimasi ini. Tombol optimasi digunakan untuk menjalankan proses optimasi menggunakan Algoritme Genetika, sedangkan *textbox* disebelah kanan merupakan tempat untuk hasil keluaran dari proses optimasi menggunakan Algoritme Genetika ini. Untuk lebih jelasnya implementasi antarmuka dapat dilihat pada Gambar 4.1.




**Gambar 4.1 Implementasi Antarmuka**

# BAB 5 PENGUJIAN DAN ANALISIS

Pada bab 5 pengujian ini akan menjelaskan tentang pengujian data dari proses yang telah dibuat, nilai-nilai parameter yang ada akan sangat berpengaruh terhadap hasil solusi. Nilai-nilai parameter terbaik akan memberikan himpunan solusi yang baik pula pada Algoritme Genetika ini, pada pengujian ini juga akan dilakukan pengujian *random injection*.

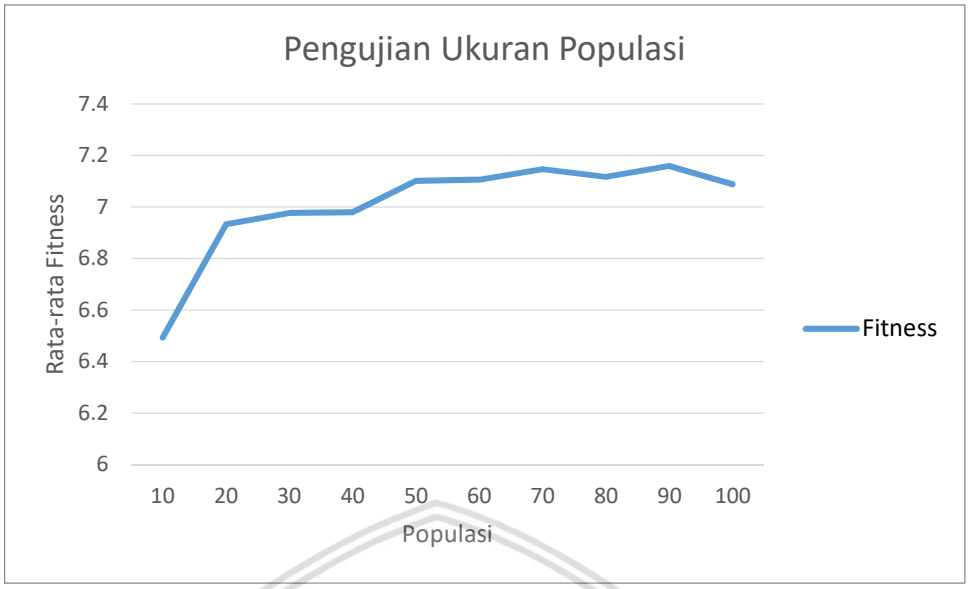
## 5.1 Pengujian Ukuran Populasi dan Analisis

Dalam pengujian ini memiliki tujuan untuk mencari ukuran populasi terbaik, hasil dari pengujian ini akan digunakan sebagai parameter ukuran populasi pada pengujian selanjutnya agar didapatkan jumlah produksi yang optimal. Ukuran populasi yang digunakan pada pengujian ini merupakan kelipatan 10 dimulai dari 10 sampai dengan 100. Untuk parameter lainya ditetapkan antara lain nilai *cr* = 0.5, *mr* = 0.5, dan maksimal generasi = 1000. Masing-masing kondisi akan dilakukan 5 kali proses percobaan, lalu diambil nilai rata-rata nya untuk diambil satu nilai tertinggi sebagai hasil ukuran populasi yang terbaik. Berikut adalah tabel perancangan pengujian ukuran populasi yang ditunjukkan pada Tabel 5.1.

**Tabel 5.1 Pengujian Ukuran Populasi**

| Ukuran Populasi | Percobaan ke - | | | | | Rata-rata |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| 10 | 5.8718 | 6.8333 | 6.2816 | 6.7231 | 6.76 | 6.49396 |
| 20 | 7.3442 | 6.6205 | 7.1689 | 7.0278 | 6.5073 | 6.93374 |
| 30 | 6.7887 | 6.9954 | 7.0488 | 7.0005 | 7.0532 | 6.97732 |
| 40 | 7.1992 | 7.0324 | 6.7355 | 6.8835 | 7.0466 | 6.97944 |
| 50 | 7.1479 | 7.1224 | 6.9086 | 7.2101 | 7.1169 | 7.10118 |
| 60 | 7.2079 | 7.1044 | 7.15 | 7.1379 | 6.929 | 7.10584 |
| 70 | 7.135 | 7.2467 | 7.0335 | 7.1636 | 7.1585 | 7.14746 |
| 80 | 7.0413 | 7.299 | 7.2073 | 6.9857 | 7.0554 | 7.11774 |
| 90 | 7.2203 | 7.0602 | 7.1972 | 7.1559 | 7.1646 | 7.15964 |
| 100 | 7.2439 | 7.0856 | 7.039 | 7.093 | 6.9808 | 7.08846 |

Berdasarkan Tabel 5.1 hasil pengujian ukuran populasi menghasilkan nilai *fitness* yang berbeda dari setiap masukan ukuran populasi yang ada. Ukuran populasi terbaik didapatkan dengan masukan 90 populasi, perbedaan masukan ukuran populasi menghasilkan nilai *fitness* yang berbeda pula. Untuk lebih mempermudah melihat perbedaan nilai nya serta mengkaji hasil pengujian nya, isi dari tabel diatas akan disajikan dalam bentuk grafik seperti pada Gambar 5.1.




**Gambar 5.1 Pengujian Ukuran Populasi**

Berdasarkan Gambar 5.1 terlihat jelas bahwa masukan ukuran populasi sangat mempengaruhi nilai *fitness* dari setiap individu. Ukuran populasi terbaik terdapat pada populasi sejumlah 90, dengan rata-rata nilai *fitness* sebesar7.15964. Sedangkan untuk nilai *fitness* terkecil terdapat pada ukuran populasi sejumlah 10, dengan rata-rata nilai *fitness* sebesar 6.49396. Dengan demikian ukuran populasi sangat mempengaruhi nilai *fitness*, semakin besar ukuran populasi akan memberikan nilai *fitness* yang terbaik pula. Dari grafik pengujian diatas terlihat jelas peningkatan nilai *fitness*, ini disebabkan karna semakin besar populasi akan memberikan ruang pencarian yang lebih luas juga. Sehingga Algoritme Genetika dapat memaksimalkan pencarian himpunan solusi yang terbaik dengan ruang pencarian yang lebih luas ini. Dengan adanya *random injection* pada setiap 10 generasi juga memberikan keragaman individu yang lebih baik, sehingga Algoritme Genetika dapat menjelejahi ruang pencarian lebih luas lagi.

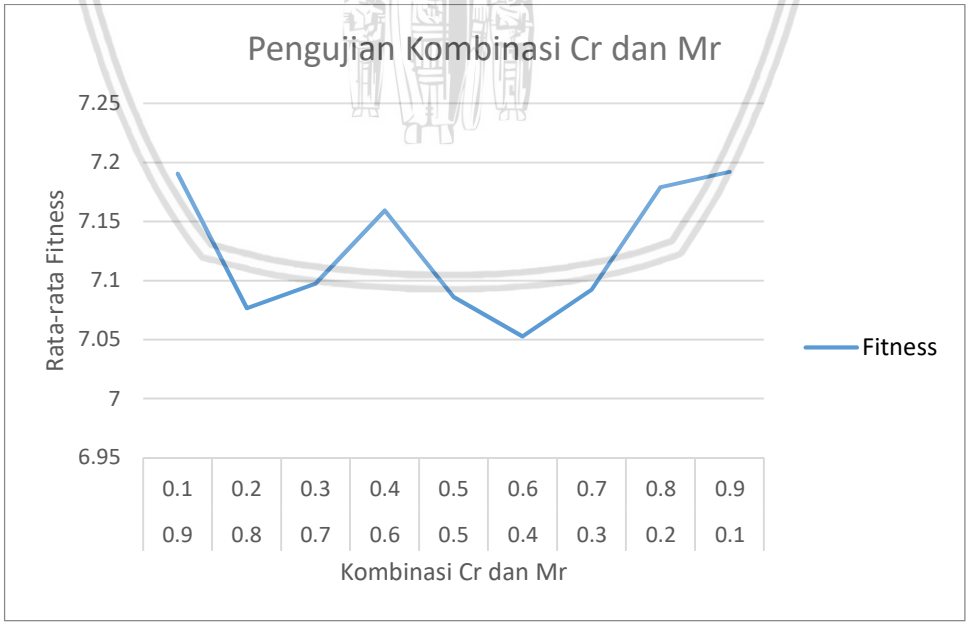
## 5.2 Pengujian Kombinasi *cr mr* dan Analisis

Dalam pengujian ini bertujuan untuk mencari kombinasi *cr* dan *mr* terbaik, hasil dari pengujian ini akan digunakan sebagai parameter kombinasi *cr* dan *mr* pada pengujian selanjutnya agar didapatkan jumlah produksi yang optimal. Kombinasi *cr* dan *mr* yang digunakan pada pengujian ini merupakan kelipatan 0,1 antara 0 dan 1. Untuk nilai *cr* dimulai dari 0.9-0.1, dan nilai *mr* dimulai dari 0.1-0.9. Untuk parameter ukuran populasi terbaik didapatkan dari pengujian ukuran populasi seperti pada Subbab 5.1, dan maksimal generasi = 1000. Masing-masing kondisi ini akan dilakukan 5 kali proses percobaan, lalu diambil nilai rata-rata nya untuk diambil satu nilai tertinggi sebagai kombinasi *cr* dan *mr* yang terbaik. Berikut adalah tabel perancangan pengujian kombinasi *cr* dan *mr* yang ditunjukkan pada Tabel 5.2.

**Tabel 5.2 Pengujian Kombinasi *cr* dan *mr***

| Kombinasi | | Percobaan ke - | | | | | Rata-rata |
|---|---|---|---|---|---|---|---|
| *cr* | *mr* | 1 | 2 | 3 | 4 | 5 | |
| 0.9 | 0.1 | 7.1558 | 7.1963 | 7.0297 | 7.3442 | 7.227 | 7.1906 |
| 0.8 | 0.2 | 7.253 | 7.1302 | 6.9739 | 6.9829 | 7.044 | 7.0768 |
| 0.7 | 0.3 | 7.1509 | 7.0618 | 7.2515 | 7.1744 | 6.8491 | 7.09754 |
| 0.6 | 0.4 | 7.2959 | 6.9738 | 7.2294 | 7.0551 | 7.2429 | 7.15942 |
| 0.5 | 0.5 | 7.0481 | 7.084 | 7.0989 | 7.1486 | 7.0517 | 7.08626 |
| 0.4 | 0.6 | 7.1092 | 7.0985 | 6.9273 | 7.088 | 7.0413 | 7.05286 |
| 0.3 | 0.7 | 7.1586 | 6.9766 | 7.1605 | 7.1346 | 7.0321 | 7.09248 |
| 0.2 | 0.8 | 7.2529 | 6.9636 | 7.2897 | 7.2933 | 7.0961 | 7.17912 |
| 0.1 | 0.9 | 7.3469 | 7.1561 | 7.203 | 7.1675 | 7.0879 | 7.19228 |

Berdasarkan Tabel 5.2 hasil pengujian kombinasi *cr* dan *mr* menghasilkan nilai *fitness* yang berbeda dari setiap masukan kombinasi *cr* dan *mr* yang ada. Pada pengujian kombinasi *cr* dan *mr* ini parameter ukuran populasi yang digunakan adalah 90, nilai tersebut didapatkan dari pengujian ukuran populasi sebelumnya. Untuk lebih mempermudah melihat perbedaan nilai nya serta mengkaji hasil pengujian nya, isi dari tabel diatas akan disajikan dalam bentuk grafik seperti pada Gambar 5.2.




**Gambar 5.2 Pengujian Kombinasi *cr* dan *mr***

Berdasarkan Gambar 5.2 terlihat jelas bahwa kombinasi *cr* dan *mr* terbaik menggunakan *cr*=0.1 dan *mr*=0.9, dengan rata-rata nilai *fitness* sebesar 7.19228. pengujian ini menggunakan parameter ukuran populasi terbaik dari pengujian

sebelumnya yang telah dilakukan, yaitu sejumlah 90 populasi. Sedangkan untuk nilai *fitness* terkecil didapatkan *cr*=0.6 dan *mr*=0.4, dengan rata-rata nilai *fitness* sebesar 7.05286. Pada permasalahan ini nilai *cr* terbaik didapatkan terlalu rendah, ini bisa disebabkan karena penggunaan batas permintaan terendah dan permintaan tertinggi. Nilai *cr* yang besar tentunya akan menghasilkan banyak anak dari reproduksinya, namun ini akan memungkinan menghasilkan banyak anak diluar batas permintaan terendah dan permintaan tertinggi dikarenakan metode *extended intermediate crossover* menggunakan nilai alpha dalam rentang -0.25 hingga 1.25 pada prosesnya. Untuk lebih jelas dapat dilihat pada Tabel 5.3.

**Tabel 5.3 Contoh Hasil Reproduksi Diluar Batas**

| Gen1 | Gen2 | Gen3 |
|---|---|---|
| 338 | 3314 | 2066 |
| 136 | 2986 | 8669 |
| 343 | 2660 | 529 |
| 412 | 4133 | 4298 |
| 374 | 4272 | 5404 |
| 190 | 404- | 2653 |
| 263 | 4107 | 1771 |
| 131 | 2285 | 9707 |
| 434 | 2190 | 2352 |
| 398 | 3507 | 8226 |
| 302 | 3544 | 8183 |
| 293 | 2475 | 1712 |
| 329 | 2303 | 9436 |
| 421 | 4336 | 5137 |
| 270 | 4496 | 3441 |
| 340 | 4054 | 2034 |
| 131 | 3984 | 6968 |
| 236 | 3338 | 5353 |

Pada Tabel 5.3 tersebut pengujian menggunakan kombinasi *cr*=0.7 dan *mr*= 0.3 terdapat 2 individu yang berada diluar batas permintaan tertinggi pada gen ke 3 yang batasnya adalah 9000. Sehingga gen tersebut akan langsung tereleminasi jauh dikarenakan melebihi batas permintaan pasar. Untuk nilai *mr* yang terlalu tinggi ini bisa disebabkan karena proses mutasi yang melakukan *random* ulang nilai pada gen terpilih dari individu sesuai batas permintaan terendah dan permintaan tertinggi, sehingga individu hasil mutasi belum tentu terbuang pada

proses seleksi dan memungkinkan memberikan himpunan solusi terbaik lebih banyak. Sedangkan untuk *mr* yang terlalu rendah akan memberikan himpunan solusi yang berada dalam rentang lebih sedikit. Dengan demikian, kombinasi *cr* dan *mr* yang cocok berperan penting sesuai dengan permasalahan yang ada. Setiap permasalahan memiliki karakteristik nya masing-masing, sehingga kombinasi *cr* dan *mr* ini belum tentu cocok dengan permasalahan menggunakan Algoritme Genetika lainnya.

## 5.3 Pengujian Jumlah Generasi dan Analisis

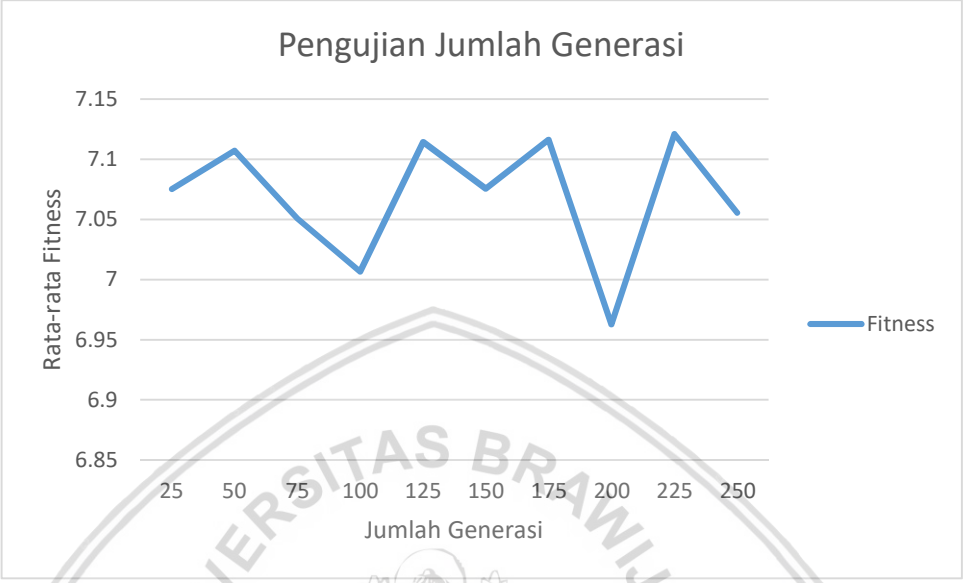
Dalam pengujian ini bertujuan untuk mencari jumlah generasi terbaik serta menggabungkan seluruh parameter terbaik sehingga didapatkan solusi akhir jumlah produksi yang optimal. Jumlah generasi yang digunakan pada pengujian ini merupakan kelipatan 25 dimulai dari 25 sampai dengan 250. Untuk parameter ukuran populasi, digunakan nilai ukuran populasi yang didapatkan dari pengujian ukuran populasi pada Subbab 5.1. Untuk parameter kombinasi *cr* dan *mr*, digunakan nilai kombinasi *cr* dan *mr* yang didapatkan dari pengujian kombinasi *cr* dan *mr* pada Subbab 5.2. Masing-masing kondisi akan dilakukan 5 kali proses percobaan, lalu diambil nilai rata-rata nya untuk diambil satu nilai tertinggi sebagai jumlah generasi yang terbaik. Hasil terbaik dari pengujian ini adalah hasil solusi akhir dari proses optimasi, hasil tersebut merupakan keluaran jumlah produksi yang optimal dari penggunan stok sisa bahan baku yang ada. Berikut adalah tabel perancangan pengujian jumlah generasi yang ditunjukkan pada Tabel 5.4.

**Tabel 5.4 Pengujian Jumlah Generasi**

| Jumlah Generasi | Percobaan ke - | | | | | Rata-rata |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| 25 | 7.2 | 6.9283 | 7.0892 | 7.1343 | 7.0248 | 7.07532 |
| 50 | 7.1799 | 6.812 | 7.1459 | 7.0882 | 7.3102 | 7.10724 |
| 75 | 7.0643 | 6.8963 | 6.9236 | 7.2464 | 7.1237 | 7.05086 |
| 100 | 7.1128 | 6.8536 | 7.1179 | 7.0295 | 6.9199 | 7.00674 |
| 125 | 7.1747 | 7.0663 | 7.0285 | 7.1031 | 7.1997 | 7.11446 |
| 150 | 6.936 | 6.9887 | 7.1295 | 7.074 | 7.2505 | 7.07574 |
| 175 | 7.2327 | 7.2099 | 7.0367 | 7.1235 | 6.9796 | 7.11648 |
| 200 | 7.1314 | 6.7687 | 7.134 | 6.8108 | 6.9686 | 6.9627 |
| 225 | 7.1305 | 7.0882 | 7.1074 | 7.0901 | 7.1901 | 7.12126 |
| 250 | 7.0575 | 7.0531 | 7.2022 | 7.0131 | 6.952 | 7.05558 |

Berdasarkan Tabel 5.4 hasil pengujian generasi ini merupakan gabungan parameter terbaik dari ukuran populasi serta kombinasi *cr* dan *mr*. Setiap pengujian jumlah generasi ini memberikan nilai *fitness* yang berbeda-beda. Untuk

lebih mempermudah melihat perbedaan nilai nya serta mengkaji hasil pengujian nya, isi dari tabel diatas akan disajikan dalam bentuk grafik seperti pada Gambar 5.3.




**Gambar 5.3 Pengujian Jumlah Generasi**

Berdasarkan Gambar 5.3 setelah menggabungkan parameter terbaik dari pengujian sebelumnya, didapatkan jumlah generasi terbaik didapatkan menggunakan generasi sebanyak 225 dengan rata-rata nilai *fitness* sebesar 7.12126 menggunakan populasi sejumlah 90 serta nilai *cr*=0.1 dan *mr*=0.9. Pada pengujian ini terlihat jelas bahwa belum tentu semakin banyak generasi akan memberikan hasil yang baik pula, ini disebabkan oleh penggunaan nilai *random* pada permasalahan Algoritme Genetika ini. Sehingga walaupun menggunakan generasi yang tidak terlalu besar, bisa jadi solusi yang akan diberikan dapat tercapai konvergen terlebih dahulu. Untuk mencegah konvergensi dini tersebut, digunakan metode *random injection* untuk menjaga keragaman dari populasi agar pada proses reproduksi dapat menghasilkan individu yang terbaik. Dengan menggunakan *random injection* ini, terlihat jelas tidak ada perubahan yang signifikan pada nilai *fitness* dari himpunan solusi pada pengujian jumlah generasi ini. Hasil gabungan dari parameter terbaik dengan ukuran populasi=90, kombinasi cr=0.1 dan mr=0.9, serta jumlah generasi=225 ini memberikan hasil solusi yang nilai *fitness* nya hanya berada antara 7.0882 hingga 7.1901.

Dikarenakan pengujian ini merupakan gabungan dari pengujian dengan semua parameter terbaik yang ada, maka dari hasil pengujian ini juga akan diambil hasil solusi dari permasalahan optimasi jumlah produksi metal roof. Untuk mendapatkan hasil solusi nya, maka akan diambil individu dengan *fitness* terbaik dari 5 percobaan yang dilakukan pada jumlah generasi 225 ini. Individu yang memiliki *fitness* terbaik dari percobaan ini adalah individu percobaan ke-5 yaitu [72, 276, 6692] dengan nilai *fitness* sebesar 7.1901. Individu tersebut mewakili

nilai asli dari produk yang ada, berarti dari hasil optimasi ini yang akan memberikan keuntungan terbaik dari penggunaan sisa bahan baku bulan februari 2017 adalah 72 meter spandek, 276 lembar zigzag, dan 6692 lembar zigzag pasiran.

## 5.4 Pengujian *Random Injection* dan Analisis

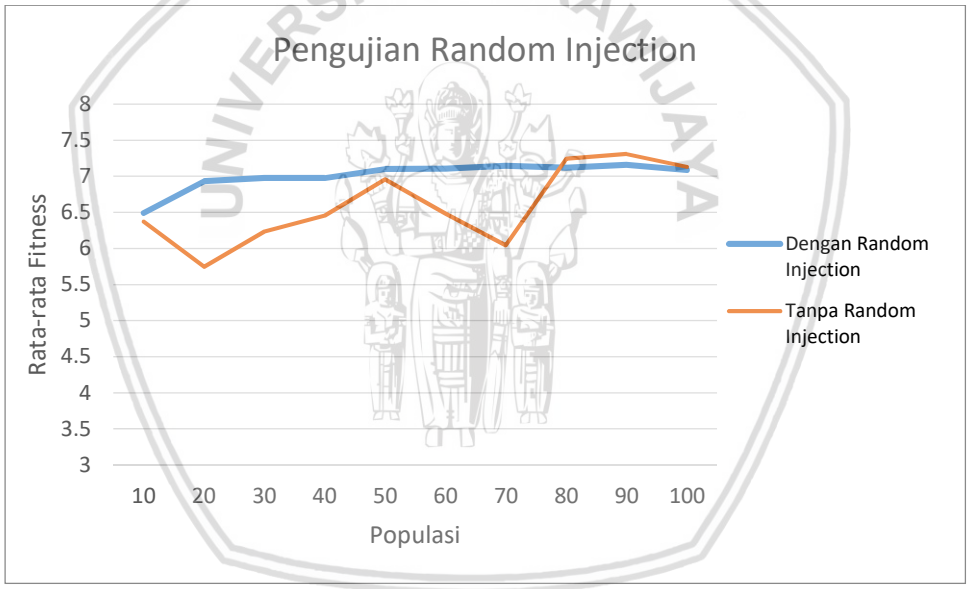
Dalam pengujian ini bertujuan untuk membandingkan hasil optimasi yang dilakukan pencegahan konvergensi dini dengan hasil optimasi yang tidak dilakukan pencegahan konvergensi dini menggunakan *random injection*. Pada pengujian ini akan dibandingkan pengujian ukuran populasi yang menggunakan *random injection* dengan pengujian ukuran populasi yang tidak menggunakan *random injection*. Karena pada subbab 5.1 telah dilakukan pengujian ukuran populasi menggunakan *random injection,* maka pada subbab ini akan dilukan pengujian ukuran populasi tanpa menggunakan random injection. Pada pengujian ini ukuran populasi yang digunakan sama dengan pengujian ukuran populasi menggunakan random injection yaitu kelipatan 10 dimulai dari 10 sampai dengan 100. Untuk parameter kombinasi *cr* dan *mr*, digunakan nilai kombinasi *cr*=0.5 dan *mr*=0.5. Jumlah generasi yang digunakan sebesar 1000 dan masing-masing kondisi akan dilakukan 5 kali proses percobaan. Berikut adalah tabel pengujian tanpa *random injection* seperti pada Tabel 5.5.

**Tabel 5.5 Pengujian Tanpa *Random Injection***

| Ukuran Populasi | Percobaan ke - | | | | | Rata-rata |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| 10 | 5.8966 | 5.5032 | 7.1511 | 6.2574 | 7.0598 | 6.37359 |
| 20 | 5.6136 | 6.3995 | 5.9281 | 3.8558 | 6.9478 | 5.74896 |
| 30 | 7.2345 | 5.6213 | 5.2442 | 7.2263 | 5.8595 | 6.23713 |
| 40 | 6.5101 | 6.8461 | 6.7075 | 5.1121 | 7.1095 | 6.45704 |
| 50 | 7.3025 | 7.0917 | 6.2602 | 7.0666 | 7.0838 | 6.960955 |
| 60 | 6.7089 | 6.099 | 6.626 | 6.0718 | 6.9213 | 6.48537 |
| 70 | 6.81 | 5.3269 | 5.67 | 6.3056 | 6.1183 | 6.04614 |
| 80 | 7.299 | 7.0425 | 7.3264 | 7.3347 | 7.2248 | 7.245475 |
| 90 | 7.2713 | 7.3407 | 7.3288 | 7.2744 | 7.3269 | 7.308395 |
| 100 | 7.158 | 6.9986 | 7.0637 | 7.2474 | 7.1908 | 7.13169 |

Berdasarkan Tabel 5.5 hasil pengujian yang sudah dilakukan dalam 1000 generasi terlihat jelas masih terdapat nilai yang konvergensi dini yaitu nilai *fitness* yang belum optimal keluar menjadi hasil solusi dari optimasi ini. Pada ukuran populasi sebesar 20 percobaan ke-4 nilai *fitness* nya adalah sebesar 3.8558, namun dari 5 percobaan pada ukuran populasi 20 tersebut nilai *fitness* yang lainnya

berada dalam rentang 5.6 hingga 6.9. Nilai *fitness* percobaan ke-4 tersebut berada dalam rentang yang terlalu jauh dengan nilai *fitness* percobaan yang lainnya. Jadi, pada kondisi ini percobaan ke-4 mengalami konvergensi dini yang dimana individu pada populasi sudah bernilai sama semua. Sehingga, proses reproduksi pada Algoritme Genetika tidak dapat melebarkan pencariannya dan akan memberikan anak yang sama dengan induknya. Kondisi ini tidak saja terjadi pada ukuran populasi 20, pada ukuran populasi 70 juga terdapat konvergensi dini. Hal ini bisa dilihat dari rata-rata nilai *fitness* nya yang mengalami penurunan, padahal rata-rata nilai *fitness* setelah ukuran populasi setelah 70 terus mengalami peningkatan dari ukuran populasi sebelumnya. Dan rata-rata nilai *fitness* pada ukuran populasi 70 juga lebih kecil dari pada ukuran populasi 30 yang seharusnya ruang pencarian dengan ukuran populasi tersebut seharusnya lebih luas dan didapatkan hasil yang lebih baik. Hasil pengujian yang menggunakan *random injection* terdapat pada Tabel 5.5 dan hasil pengujian yang tidak menggunakan *random injection* terdapat pada Tabel 5.1. Untuk melihat perbandingan yang lebih jelas dari hasil yang menggunakan *random injection* dan yang tidak menggunakan *random injection* dapat dilihat pada Gambar 5.4.




**Gambar 5.4 Pengujian Random Injection**

Pada hasil pengujian ini terlihat jelas bahwa hasil tanpa menggunakan *random injection* rata-rata nilai *fitness* nya naik turun, seperti pada ukuran populasi 20 yang lebih rendah dari ukuran populasi 10. Dengan ukuran populasi yang lebih besar seharusnya Algoritme Genetika dapat lebih memperluas pencariannya, namun pada kondisi ini terdapat konvergensi dini sehingga hasil yang diberikan belum optimal walaupun generasi yang digunakan sejumlah 1000. Sehingga dengan menggunakan random injection pada subbab sebelumnya memberikan nilai fitness yang lebih baik, *random injection* mampu memberikan keragaman individu pada populasi sehingga sangat membantu Algoritme Genetika untuk menjelejahi ruang pencarian yang lebih luas untuk menemukan hasil yang mendekati optimal. Dan pada pengujian menggunakan *random injection* setiap

percobaanya tidak terdapat nilai *fitness* yang berada dalam rentang terlalu jauh atau terjadinya konvergensi dini.

# BAB 6 PENUTUP

## 6.1 Kesimpulan

Berdasarkan hasil dari keseluruhan penelitian pengoptimalan jumlah produksi metal roof dengan menggunakan Algoritme Genetika pada PT.Comtech Metalindo Terpadu serta merujuk dari permasalahan yang ada, dapat disimpulkan bahwa:

1. Terdapat 3 parameter yang digunakan pada penelitian ini untuk mendapatkan hasil yang mendekati optimal, yaitu ukuran populasi, kombinasi *cr* dan *mr*, serta jumlah generasi. Pengujian setiap parameter dilakukan sebanyak 5 kali percobaan, untuk mendapatkan rata-rata yang terbaik. Berdasarkan pengujian yang telah dilakukan pada Bab 5, hasil pengujian parameter yang mendekati optimal didapatkan jumlah populasi sebesar 90, *cr*=0.1 dan *mr*=0.9, serta jumlah generasi sebesar 225 dengan rata-rata nilai *fitness* 7.12126.
2. *Random injection* merupakan suatu metode untuk menangani konvergensi dini. Konvergensi dini merupakan suatu kondisi dimana individu pada populasi hasil hampir bernilai sama semua, sehingga akan menyebabkan tidak adanya perubahan hasil solusi yang akan diberikan walaupun solusi tersebut belum mendekati optimal. Apabila kondisi tersebut terjadi tentunya Algoritme Genetika tidak mampu mengembangkan pencarian lebih jauh. Pada permasalahan ini *random injection* mampu memberikan keragaman individu pada populasi, sehingga sangat membantu Algoritme Genetika untuk menjelejahi ruang pencarian yang lebih luas untuk menemukan hasil yang mendekati optimal. Berdasarkan pengujian parameter menggunakan *random injection* pada bab 5, terlihat hasil solusi memberikan nilai *fitness* yang memiliki rentang tidak terlalu jauh dan tidak terdapat solusi yang nilai *fitness* nya terlalu rendah. Dengan adanya *random injection* ini mampu memberikan kesempatan pada individu lainnya untuk bersaing, sehingga tidak terdapat nilai yang konvergen diawal dengan nilai *fitness* yang rendah.

## 6.2 Saran

Berdasarkan hasil dari keseluruhan penelitian yang telah dilakukan ini, peneliti memberikan saran untuk penelitian selalnjutnya. Adapun saran yang dapat peneliti berikan adalah sebagai berikut:

1. Pada penelitian ini metode *crossover* yang digunakan memungkinkan memberikan hasil yang diluar batas permintaan terendah dan permintaan tertinggi, sedangkan batas tersebut dibuat agar sesuai dengan permintaan pasar. Oleh karena itu, untuk penelitian selanjutnya dapat digunakan metode yang berbeda untuk dibandingkan metode manakah yang lebih baik.
2. Dapat menggabungkan atau memodifikasi Algoritme Genetika dengan Algoritme lainnya, agar didapatkan perbaikan yang lebih baik dan menghasilkan hasil optimasi yang lebih optimum.

# DAFTAR PUSTAKA


Black, E. P., 2001. Dictionary of Algorithms, Data Structures, and Problems.

Fitrianur, K. N., Putri, R. R. M. & Wicaksono, S. A., 2017. Optimasi Peningkatan Laba Produksi Abon dengan Menggunakan Algoritma Genetika (Studi Kasus UKM Poklahsar Berkah Lumintu Tulungagung). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 2(5), pp. 1883-1893.

Ilmi, R. R., Mahmudy, W. F. & Ratnawati, E. D., 2015. OPTIMASI PENJADWALAN PERAWAT MENGGUNAKAN. *DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya,* 5(13).

Kusumadewi, S., 2003. *Artificial Intelligence (Teknik dan Aplikasinya).* Yogyakarta: Graha Ilmu.

Mahmudy, W. F., 2013. *Algoritma Evolusi.* Malang: Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.

Mahmudy, W. F., Marian, R. & Luong, L., 2014. Hybrid Genetic Algorithms For Part Type Selection And Machine Loading Problems With Alternative Production Plans In Flexible Manufacturing System. *ECTI Transactions on Computer and Information Technology (ECTI- CIT),* 8(1), pp. 80-93.

Mahmudy, W. F., Marian, R. M. & Luong, L. H. S., 2013a. *Optimization of part type selection and loading problem with alternative production plans in flexible manufacturing system using hybrid genetic algorithms – Part 1: modelling and representation.* Chonburi, Thailand, 31 Jan - 1 Feb, 5th International Conference on Knowledge and Smart Technology (KST), pp. 75-80.

Mahmudy, W. F., Marian, R. M. & Luong, L. H. S., 2013b. *Optimization of part type selection and loading problem with alternative production plans in flexible manufacturing system using hybrid genetic algorithms – Part 2: genetic operators & results.* Chonburi, Thailand, 31 Jan - 1 Feb, 5th International Conference on Knowledge and Smart Technology (KST), pp. 81-85.

Muhlenbein, H. & Schlierkamp-Voosen, D., 1993. Predictive models for the breeder genetic algorithm; continuous parameter optimization. *Evolutionary Computation,* Volume 1, pp. 25-49.

Saputra, E. H., 2018. *Industri Manufaktur Menggeliat.* [Online] Available at: http://mediaindonesia.com/read/detail/145546-industri-manufaktur-menggeliat
[Accessed 18 April 2018].

Situmorang, A., 2008. *Ekonomi Jilid I untuk SMA/MA Kelas X.* Jakarta: ESIS.

Suhartono, E., 2015. Optimasi Penjadwalan Mata Kuliah dengan Algoritma Genetika (Studi Kasus di AMIK JTC Semarang). *INFOKAM.*

Sutojo, T., Mulyanto, E. & Suhartanto, V., 2011. *Kecerdasan Buatan.* Yogyakarta: ANDI.

Yogeswaran, M., Ponnambalam, S. G. & Tiwari, M. K., 2009. 'An efficient hybrid evolutionary heuristic using genetic algorithm and simulated annealing algorithm to solve machine loading problem in FMS'. *International Journal of Production Research,* 47(19), pp. 5421-5448.